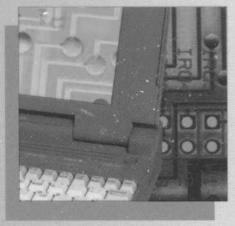
MAY 1997



AVR®

ENHANCED RISC MICROCONTROLLER DATA BOOK

ATMEL

Atmel
An ISO 9000 Registered Company

AVR®

Atmel Corporation
8-Bit RISC Microcontrollers
Data Book
May 1997

Preliminary

Important Notice

Atmel guarantees that its circuits will be free from defects of material and workmanship under normal use and service, and that these circuits will perform to current specifications in accordance with, and subject to, the Company's standard warranty which is detailed in Atmel's Purchasing Order Acknowledgment.

Atmel reserves the right to change devices or specifications detailed in this data book at any time without notice, and assumes no responsibility for any errors within this document. Atmel does not make any commitment to update this information. Atmel assumes no responsibility for the use of any circuits described in this data book, nor does the Company assume responsibility for the functioning of undescribed features or parameters.

In the absence of a written agreement to the contrary, Atmel assumes no liability with respect to the use of semiconductor devices described in this data book for applications assistance, customers' product design or infringement of patents or copyrights of third parties.

Atmel's products are not authorized for use as critical components in life support devices or systems and the use as such implies that user bears all risk of such use.

If Atmel is an approved vendor on a Standard Microcircuit Drawing (SMD), the Atmel similar part number specification is compliant with the SMD.

Printed on recycled paper.

Atmel Corporation designs, manufactures, and markets high quality and high performance CMOS memory, logic and analog integrated circuits. Founded in 1984, the Company serves the manufacturers of computation, communications and instrumentation equipment in commercial, industrial and military environments.

Atmel's broad line of products provide customers with a variety of solutions to their memory and logic applications. Atmel offers high-density, high-speed memory and logic standard products as well as custom gate arrays.

Atmel guarantees quality and reliability by fabricating all products— no matter what their intended application— to meet or exceed the specifications of Military Standard 883.

Whether you are new to programmable logic or an experienced user, Atmel is committed to your success. If you have any questions or would like to place an order, please contact your local Atmel sales office as listed in the back of this data book, or contact Atmel's corporate headquarters:

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
TEL: (408) 441-0311
FAX: (408) 487-2600

Fax-on-Demand

North America:
1-(800) 29-ATMEL
1-(800) 292-8635

International:
1-(408) 441-0732

e-mail

literature@atmel.com

Web Site

http://www.atmel.com

BBS

1-(408) 436-4309

We thank you for considering Atmel semiconductors.

Table of Contents

(continued)

Section 3 **AT90S2313** (Continued)

Section 4 **AT90S4414**

Section 9 Miscellaneous Information

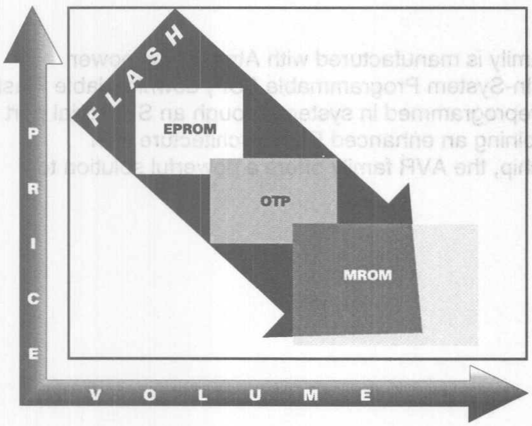
ATMEL

AVR® Enhanced RISC Microcontrollers

Atmel Corporation is a leading manufacturer of a broad range of high performance, low power nonvolatile memory and logic integrated circuits (ICs) that focus in the telecommunications, computer, networking, consumer and automotive markets. Atmel's Flash-based microcontroller families incorporate advanced single-voltage Flash memory technology in the industry's broadest line of Flash and EEPROM based products.

With the Flash memory-based microcontrollers from Atmel, you can achieve safe, easy reconfigurability:

- -Change code in seconds, shortening the development cycle
- -Stock just one part
- -Zero scrap due to misprogramming
- -Accelerate product testing
- -Make changes remotely
- -Customize each product on the line



The AVR enhanced RISC microcontrollers are based on a new RISC architecture that has been developed to take advantage of semiconductor integration and software capabilities for the 1990's. The resulting microcontrollers offer the highest MIPS/milliwatt capability available in the 8-bit MCU market.

High level languages are rapidly becoming the standard programming methodology for embedded microcontrollers due to improved time-to-market and simplified maintenance support. The AVR architecture was developed in conjunction with C language experts to ensure that the hardware and software work hand in hand to develop highly efficient, high performance code.

0855A

In order to optimize code size, performance and power consumption, the AVR architecture has incorporated a large fast-access register file and fast single-cycle instructions.

The fast-access RISC register file consists of 32 general purpose working resisters. Traditional accumulator based architectures require large amounts of program code for data transfers between the accumulator and memory. With 32 working registers (accumulators) in the AVR these data transfers are eliminated.

The AVR pre-fetches an instruction during the previous instruction execution and then executes in a single clock cycle. In other CISC- and RISC-like architectures, the external oscillator clock is divided down (by as much as 12 times) to the traditional internal execution cycle. The AVR enhanced RISC microcontrollers execute an instruction in a single clock cycle and are the first true RISC machines in the 8-bit market.

The AVR architecture supports a complete spectrum of price performance from simple small pin count controllers to high range devices with large on-chip memories. The Harvard style architecture directly addresses up to 8M bytes of program memory and 8M bytes of data memory. The register files is dual mapped and can be addresses as part of the on-chip SRAM memory to enable fast context switching.

The AVR enhanced RISC microcontroller family is manufactured with Atmel's low-power nonvolatile CMOS technology. The on-chip In-System Programmable (ISP) downloadable Flash memory allows the program memory to be reprogrammed in system through an SPI serial port or by a conventional memory programmer. By combining an enhanced RISC architecture with downloadable Flash memory on the same chip, the AVR family offers a powerful solution to embedded control applications.

ATMEL

Contents

2

8-Bit AVR® Microcontroller with 1K bytes Downloadable Flash

Features

- Utilizes the *AVR*® Enhanced RISC Architecture
- AV*R* - High Performance and Low Power RISC Architecture
- 89 Powerful Instructions - Most Single Clock Cycle Execution
- 1K bytes of In-System Reprogrammable Downloadable Flash
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 64 bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 32 x 8 General Purpose Working Registers
- 15 Programmable I/O Lines
- V_{CC}: 2.7 - 6.0V
- Fully Static Operation, 0 - 16 MHz
- Instruction Cycle Time: 62.5 ns @ 16 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes
- Programming Lock for Software Security
- 20-Pin Device
- Selectable On-Chip RC Oscillator for Zero External Components

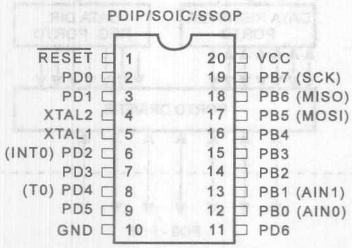
Description

The AT90S1200 is a low-power CMOS 8-bit microcontroller based on the *AVR*® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S1200 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with the 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

(continued)

Pin Configuration



0838A

Block Diagram

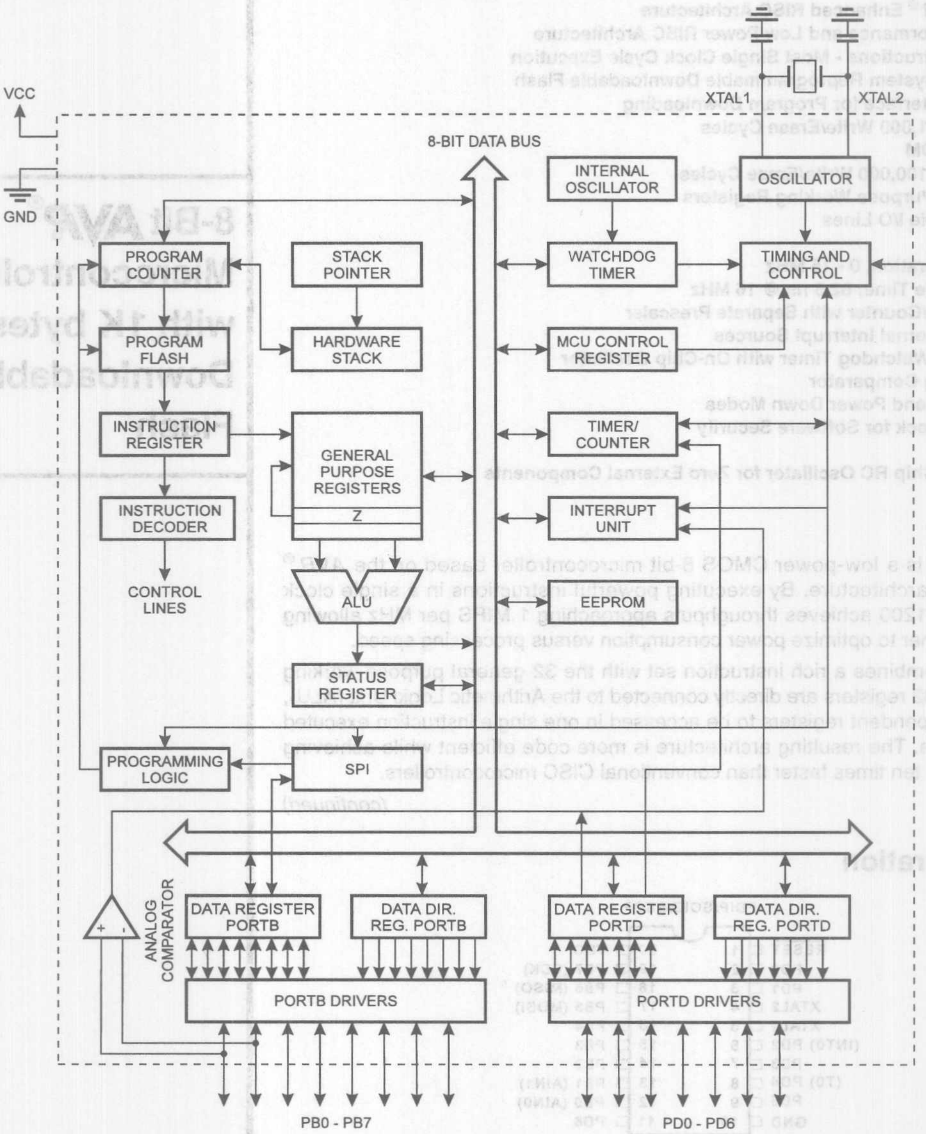


Figure 1. The AT90S1200 Block Diagram

Description (Continued)

The architecture supports high level languages efficiently as well as extremely dense assembler code programs. The AT90S1200 provides the following features: 1K bytes of Downloadable Flash, 64 bytes EEPROM, 15 general purpose I/O lines, 32 general purpose working registers, internal and external interrupts, programmable Watchdog Timer with internal oscillator, an SPI serial port for program downloading and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the registers, timer/counter, watchdog and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next external interrupt or hardware reset.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip Downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining an enhanced RISC 8-bit CPU with Downloadable Flash on a monolithic chip, the Atmel AT90S1200 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90S1200 *AVR* is supported with a full suite of program and system development tools including: macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Supply voltage pin.

GND

Ground pin.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port. Port pins can provide internal pullups (selected for each bit). PB0 and PB1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip analog comparator. The Port B output buffers can sink 20mA and can drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

Port B also serves the functions of various special features of the AT90S1200 as listed on Page 2-27.

Port D (PD6..PD0)

Port D has seven bi-directional I/O pins with internal pullups, PD6..PD0. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (IIL) if the pullups are activated.

Port D also serves the functions of various special features of the AT90S1200 as listed on Page 2-31.

$\overline{\text{RESET}}$

Reset input. A low on this pin for two machine cycles while the oscillator is running resets the device.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

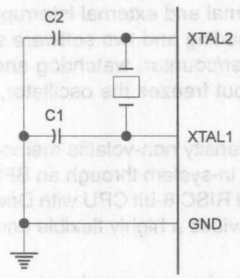


Figure 2. Oscillator Connections

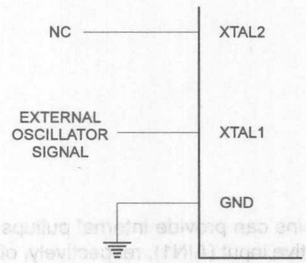


Figure 3. External Clock Drive Configuration

On-Chip RC Oscillator

An on-chip RC oscillator running at a fixed frequency of 1 MHz can be selected as the MCU clock source. If enabled, the AT90S1200 can operate with no external components. A control bit - RCEN in the Flash Memory selects the on-chip RC oscillator as the clock source when programmed ('0'). The AT90S1200 is normally shipped with this bit unprogrammed ('1'). Parts with this bit programmed can be ordered as AT90S1200A. The RCEN-bit can be changed by parallel programming only. When using the on-chip RC oscillator for serial program downloading, the RCEN bit must programmed in parallel programming mode first.

AT90S1200 *AVR* Enhanced RISC Microcontroller CPU

The AT90S1200 *AVR* RISC microcontroller is upward compatible with the *AVR* Enhanced RISC Architecture. The programs written for the AT90S1200 MCU are compatible with the range of *AVR* 8-bit MCUs (AT90Sxxxx) with respect to source code and clock cycles for execution.

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one ALU (Arithmetic Logic Unit) operation is executed. Two operands are output from the register file, the operation is executed, and the result is stored back in the register file - in one clock cycle.

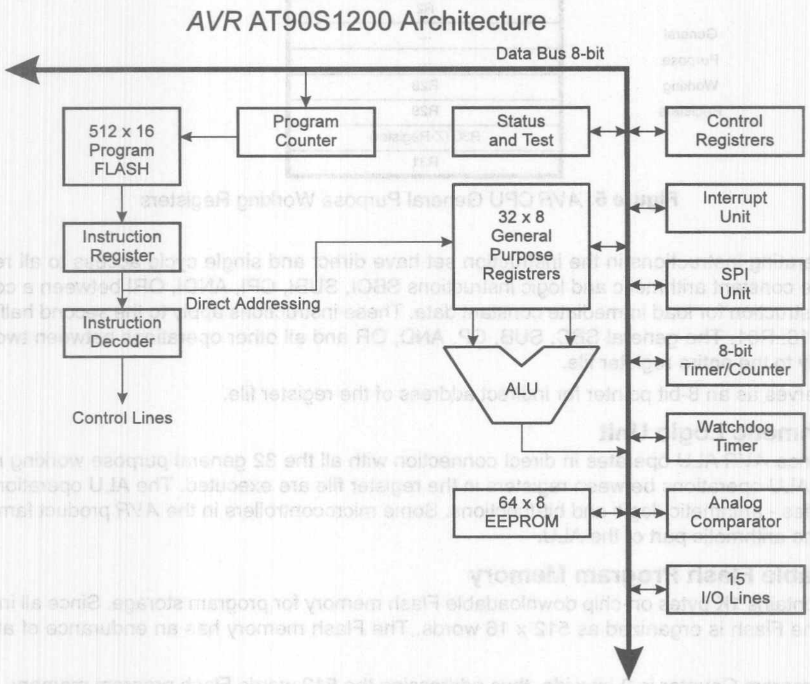


Figure 4. The AT90S1200 *AVR* Enhanced RISC Architecture

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S1200 *AVR* Enhanced RISC microcontroller architecture. The *AVR* uses a Harvard architecture concept - with separate memories and buses for program and data memories. The program memory is accessed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system downloadable Flash memory.

With the relative jump and relative call instructions, the whole 512 address space is directly accessed. All *AVR* instructions have a single 16-bit word format, meaning that every program memory address contains a single 16-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is a 3 level deep hardware stack dedicated for subroutines and interrupts.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, and other I/O functions. The memory spaces in the *AVR* architecture are all linear and regular memory maps.

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt address vector the higher priority.

The General Purpose Register File

Figure 5 shows the structure of the 32 general purpose registers in the CPU.

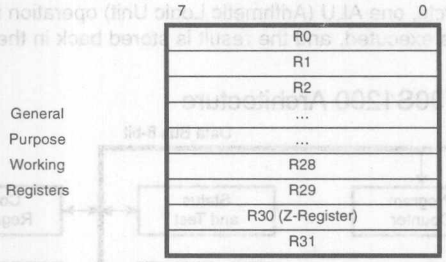


Figure 5. *AVR* CPU General Purpose Working Registers

All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file - R16..R31. The general SBC, SUB, CP, AND, OR and all other operations between two registers or on a single register apply to the entire register file.

Register 30 also serves as an 8-bit pointer for indirect address of the register file.

The ALU - Arithmetic Logic Unit

The high-performance *AVR* ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories - arithmetic, logic and bit-functions. Some microcontrollers in the *AVR* product family feature a hardware multiplier in the arithmetic part of the ALU.

The Downloadable Flash Program Memory

The AT90S1200 contains 1K bytes on-chip downloadable Flash memory for program storage. Since all instructions are single 16-bit words, the Flash is organized as 512 x 16 words. The Flash memory has an endurance of at least 1000 write/erase cycles.

The AT90S1200 Program Counter is 9-bit wide, thus addressing the 512 words Flash program memory.

See Page 2-34 for a detailed description on Flash data downloading.

The Program and Data Addressing Modes

The AT90S1200 *AVR* Enhanced RISC Microcontroller supports powerful and efficient addressing modes. This section describes the different addressing modes supported in the AT90S1200. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

REGISTER DIRECT, SINGLE REGISTER RD

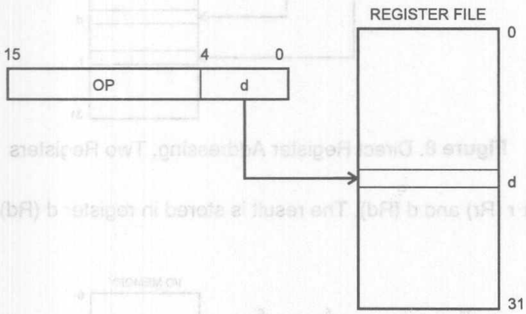


Figure 6. Direct Single Register Addressing

The operand is contained in register d (Rd).

REGISTER INDIRECT

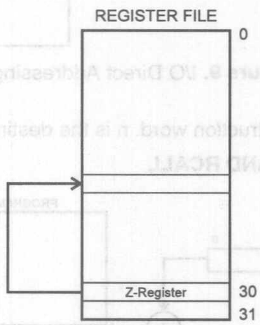


Figure 7. Indirect Register Addressing

The register accessed is the one pointed to by the Z-register (R30).

REGISTER DIRECT, TWO REGISTERS RD AND RR

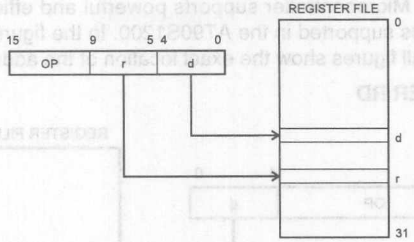


Figure 8. Direct Register Addressing, Two Registers

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O DIRECT

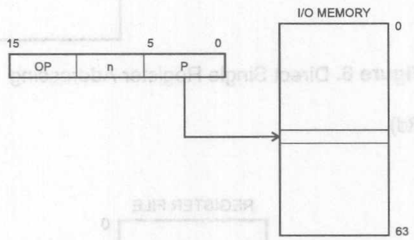


Figure 9. I/O Direct Addressing

Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

RELATIVE PROGRAM ADDRESSING, RJMP AND RCALL

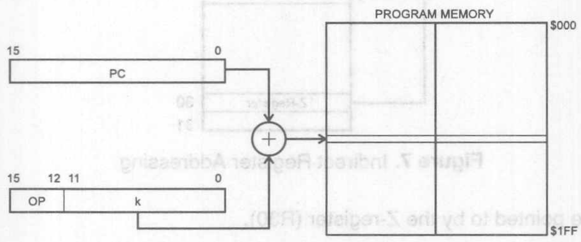


Figure 10. Relative Program Memory Addressing

Program execution continues at address PC + k. The relative address k is in the range from -2K to +(2K - 1).

Subroutine and Interrupt Hardware Stack

The AT90S1200 uses a 3 level deep hardware stack for subroutines and interrupts. The hardware stack is 9 bit wide and stores the Program Counter - PC - return address while subroutines and interrupts are executed.

RCALL instructions and interrupts push the PC return address onto stack level 0, and the data in the other stack levels 1-2 are pushed one level deeper in the stack. When a RET or RETI instruction is executed the returning PC is fetched from the stack level 0, and the data in the other stack levels 1-2 are popped one level in the stack.

If more than 3 subsequent subroutine calls or interrupts are executed, the first values written to the stack are overwritten.

The EEPROM Data Memory

The AT90S1200 contains 64 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on Page 2-24 specifying the EEPROM address register, the EEPROM data register, and the EEPROM control register. For the SPI data downloading, see Page 2-40 for a detailed description.

Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The *AVR* CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 11 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

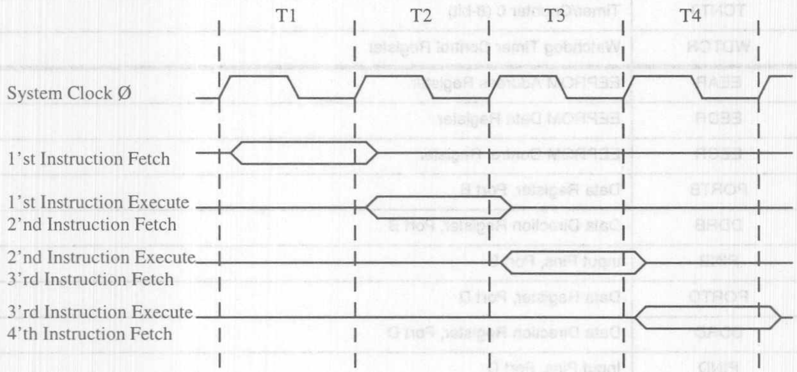


Figure 11. The Parallel Instruction Fetches and Instruction Executions

Figure 12 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

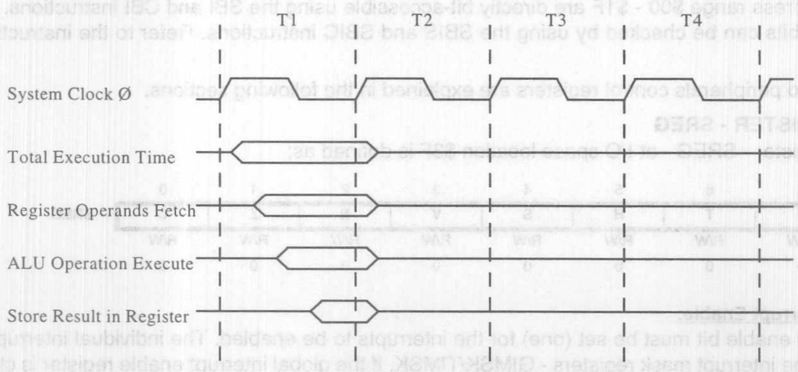


Figure 12. Single Cycle ALU Operation

I/O Memory

The I/O space definition of the AT90S1200 is shown in the following table:

Table 1. The AT90S1200 I/O Space

Address Hex	Name	Function
$3F	SREG	Status REGister
$3B	GIMSK	General Interrupt MaSK register
$39	TIMSK	Timer/Counter Interrupt MaSK register
$38	TIFR	Timer/Counter Interrupt Flag register
$35	MCUCR	MCU general Control Register
$33	TCCR0	Timer/Counter 0 Control Register
$32	TCNT0	Timer/Counter 0 (8-bit)
$21	WDTCR	Watchdog Timer Control Register
$1E	EEAR	EEPROM Address Register
$1D	EEDR	EEPROM Data Register
$1C	EECR	EEPROM Control Register
$18	PORTB	Data Register, Port B
$17	DDRB	Data Direction Register, Port B
$16	PINB	Input Pins, Port B
$12	PORTD	Data Register, Port D
$11	DDRD	Data Direction Register, Port D
$10	PIND	Input Pins, Port D
$08	ACSR	Analog Comparator Control and Status Register

Note: Reserved and unused locations are not shown in the table.

All the different AT90S1200 I/Os and peripherals are placed in the I/O space. The different I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set chapter for more details.

The different I/O and peripherals control registers are explained in the following sections.

THE STATUS REGISTER - SREG

The AVR status register - SREG - at I/O space location $3F is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - I : Global Interrupt Enable:

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in the interrupt mask registers - GIMSK/TIMSK. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled, independent of the GIMSK/TIMSK values. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

Bit 6 - T : Bit Copy Storage:
The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

Bit 5 - H : Half Carry Flag:
The half carry flag H indicates a half carry in some arithmetic operations. See the Instruction Set Description for detailed information.

Bit 4 - S : Sign Bit, S = N⊕V :
The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set Description for detailed information.

Bit 3 - V : Two's Complement Overflow Flag:
The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set Description for detailed information.

Bit 2 - N : Negative Flag:
The negative flag N indicates a negative result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 1 - Z : Zero Flag:
The zero flag Z indicates a zero result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 0 - C : Carry Flag:
The carry flag C indicates a carry in an arithmetic or logic operation. See the Instruction Set Description for detailed information.

Reset and Interrupt Handling

The AT90S1200 provides 3 different interrupt sources. These interrupts and the separate reset vector, each have a separate program vector in the program memory space. All the interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 2. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 - the External Interrupt Request 0 etc.

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
4	$002	TIMER0, OVF0	Timer/Counter0 Overflow
5	$003	ANA_COMP	Analog Comparator

The most typical and general program setup for the Reset and Interrupt Vector Addresses are:

```
Address   Labels    Code                        Comments
$000                rjmp    RESET               ; Reset handle
$001                rjmp    EXT_INT0            ; IRQ0 handle
$002                rjmp    TIM0_OVF            ; Timer0 overflow handle
$003                rjmp    ANA_COMP            ; Analog Comparator Handle
;
$004      MAIN:     <instr> xxx                 ; Main program start
...       ...       ...     ...
```

RESET SOURCES

The AT90S1200 has three sources of reset:

- Power-On Reset. The MCU is reset when a supply voltage is applied to the V_{CC} and GND pins.
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for more than two XTAL cycles
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are then set to their initial values, and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP - relative jump - instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 13 shows the reset logic. Table 3 defines the timing and electrical parameters of the reset circuitry. Note that Power On Reset timing is clocked by the internal RC oscillator. Refer to characterization datdata for RC oscillator frequency at other V_{CC} voltages.

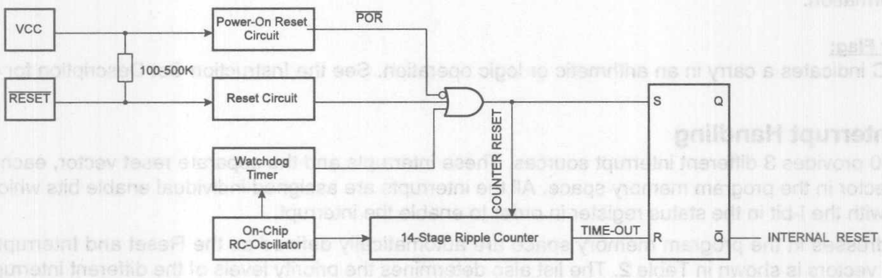


Figure 13. Reset Logic

Table 3. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-On Reset Threshold Voltage	1.8	2	2.2	V
V_{RST}	Pin Threshold Voltage		$V_{CC}/2$		V
t_{POR}	Power-On Reset Period	2	3	4	ms
t_{TOUT}	Reset Delay Time-Out Period	11	16	21	ms

POWER-ON RESET

A Power-On Reset (POR) circuit ensures that the device is not started until V_{CC} has reached a safe level. As shown in Figure 13, an internal timer clocked from the Watchdog timer oscillator prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-On Threshold voltage - V_{POT}, regardless of the V_{CC} rise time (see Figure 14 and Figure 15). The total reset period is the Power-On Reset period - t_{POR} + the Delay Time-out period - t_{TOUT}.

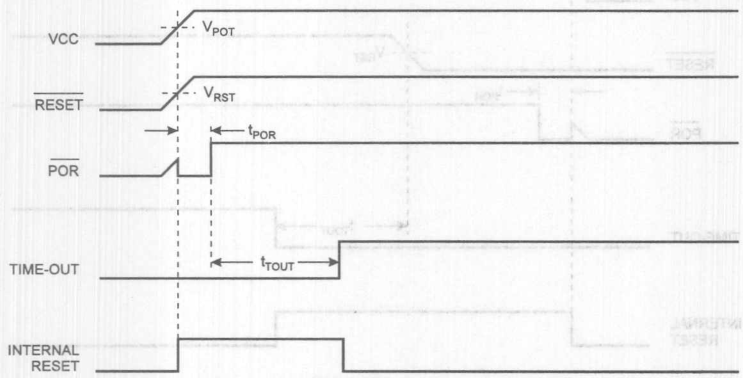


Figure 14. MCU Start-Up, RESET Tied to V_{CC} or Unconnected. Rapidly Rising V_{CC}

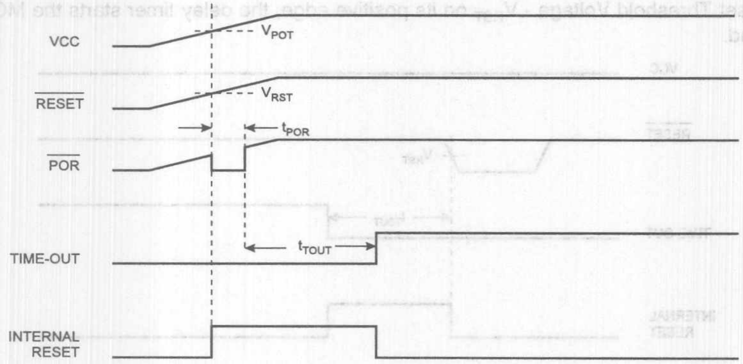


Figure 15. MCU Start-Up, RESET Tied to V_{CC} or Unconnected. Slowly Rising V_{CC}

As the $\overline{\text{RESET}}$ pin is pulled high by an on-chip resistor, the pin can be left unconnected if no external reset is required. Connecting $\overline{\text{RESET}}$ to V_{CC} will have the same effect. By holding the $\overline{\text{RESET}}$ pin low for a period after V_{CC} has been applied, the Power-On Reset period can be extended. Refer to Figure 16 for a timing example on this.

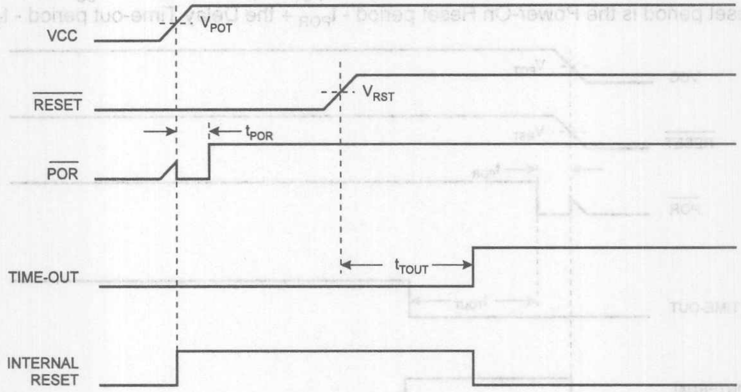


Figure 16. MCU Start-Up, $\overline{\text{RESET}}$ Controlled Externally

EXTERNAL RESET

An external reset is generated by a low level on the pin. The pin must be held low for at least two crystal clock cycles. When reaches the Reset Threshold Voltage - V_{RST} on its positive edge, the delay timer starts the MCU after the Time-out period t_{TOUT} has expired.

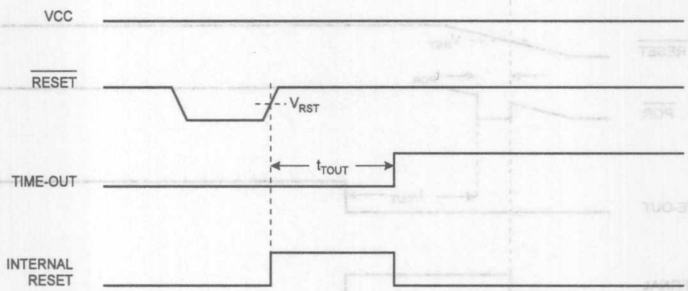


Figure 17. External Reset During Operation

WATCHDOG RESET

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT}. Refer to Page 2-23 for details on operation of the Watchdog.

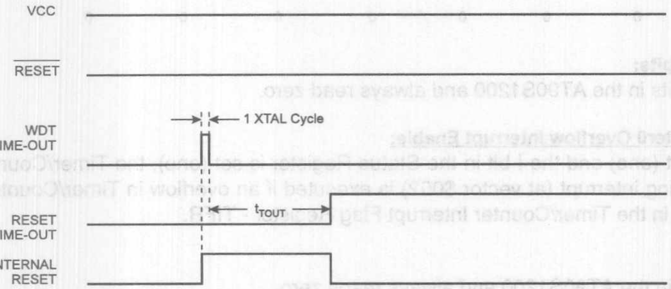


Figure 18. Watchdog Reset During Operation

INTERRUPT HANDLING

The AT90S1200 has two Interrupt Mask control registers GIMSK - General Interrupt MASK register - at I/O space address \$3B and the TIMSK - Timer/Counter Interrupt MaSK register at I/O address \$39.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable interrupts. The I-bit is set (one) when a Return from Interrupt instruction - RETI - is executed.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

THE GENERAL INTERRUPT MASK REGISTER - GIMSK

Bit	7	6	5	4	3	2	1	0	
\$3B	-	INT0	-	-	-	-	-	-	GIMSK
Read/Write	R	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - Res : Reserved bit:
This bit is a reserved bit in the AT90S1200 and always read zero.

Bit 6 - INT0 : External Interrupt Request 0 Enable:
When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control0 bit 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. INT0 can activated even if the pin is configured as an output. See also Page 2-18.

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read as zero.

THE TIMER/COUNTER INTERRUPT MASK REGISTER - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39	-	-	-	-	-	-	TOIE0	-	TIMSK
Read/Write	R	R	R	R	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..2 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read zero.

Bit 1 - TOIE0 : Timer/Counter0 Overflow Interrupt Enable:
When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt (at vector $002) is executed if an overflow in Timer/Counter0 occurs. The Overflow Flag (Timer0) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 0 - Res: Reserved bit:
This bits is a reserved bit in the AT90S1200 and always reads zero.

THE TIMER/COUNTER INTERRUPT FLAG REGISTER - TIFR

Bit	7	6	5	4	3	2	1	0	
$38	-	-	-	-	-	-	TOV0	-	TIFR
Read/Write	R	R	R	R	R	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..2 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read zero.

Bit 1 - TOV0 : Timer/Counter0 Overflow Flag:
The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

Bit 0 - Res: Reserved bit:
This bits is a reserved bit in the AT90S1200 and always reads zero.

EXTERNAL INTERRUPTS

The external interrupt is triggered by the INT0 pin. The interrupt can trigger on rising edge, falling edge or level. This is set up as described in the specification for the MCU control register - MCUCR. When INT0 is level triggered, the interrupt is pending as long as INT0 is held low.

The interrupt is triggered even if INT0 is configured as an output. This provides a way to generate a software interrupt.

The interrupt flag can not be directly accessed by the user. If an external edge triggered interrupt is suspected to be pending, the flag can be cleared as follows.

1. Disable the external interrupt by clearing the INT0 flag in GIMSK.
2. Select level triggered interrupt.
3. Select desired interrupt edge.
4. Re-enable the external interrupt by setting INT0 in GIMSK.

INTERRUPT RESPONSE TIME

The interrupt execution response for all the enabled *AVR* interrupts is 4 clock cycles minimum. After the 4 clock cycles the program vector address for the actual interrupt handling routine is executed. During this 4 clock cycle period, the Program Counter (9 bits) is pushed onto the Stack. The vector is a relative jump to the interrupt routine, and this jump takes 2 clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine takes 4 clock cycles. During these 4 clock cycles, the Program Counter (9 bits) is popped back from the Stack. When *AVR* exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register - SREG - is not handled by the AV*R* hardware, neither for interrupts nor for subroutines. For the routines requiring a storage of the SREG, this must be performed by user software.

Note that the Subroutine and Interrupt Stack is a 3-level true hardware stack, and if more than 3 nested subroutines and interrupts are executed, only the most recent 3 return addresses are stored.

THE MCU CONTROL REGISTER - MCUCR

The MCU Control Register contains general microcontroller control bits for general MCU control functions.

Bit	7	6	5	4	3	2	1	0	
$35	-	-	SE	SM	-	-	ISC01	ISC00	MCUCR
Read/Write	R	R	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7, 6 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read zero.

Bit 5 - SE : Sleep Enable:
The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

Bit 4 - SM : Sleep Mode:
This bit selects between the two available sleep modes. When SM is cleared (zero), Idle Mode is selected as Sleep Mode. When SM is set (one), Power Down mode is selected as sleep mode. For details, refer to the paragraph "Sleep Modes" below.

Bits 3, 2 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read zero.

Bits 1, 0 - ISC01, ISC00 : Interrupt Sense Control 0 bit 1 and bit 0:
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask in the GIMSK register is set. The level and edges on the external INT0 pin that activate the interrupt are defined as:

Table 4. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Note: When changing the ISC10/ISC00 bits, INT0 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Sleep Modes

To enter the sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file and the I/O memory are unaltered. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset vector.

Note that if a *level* triggered interrupt is used for wake-up from power down, the low level must be held for a time longer than the oscillator start-up time of 16 ms. Otherwise, the interrupt flag may return to zero before the MCU starts executing.

IDLE MODE

When the SM bit is cleared (zero), the SLEEP instruction forces the MCU into the Idle Mode stopping the CPU but allowing Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like Timer Overflow interrupt and watchdog reset. If wakeup from the Analog Comparator interrupt is not required, the analog comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status register - ACSR. This will reduce power consumption during Idle Mode.

POWER DOWN MODE

When the SM bit is set (one), the SLEEP instruction forces the MCU into the Power Down Mode. In this mode, the external oscillator is stopped. The user can select whether the watchdog shall be enabled during power-down mode. If the watchdog is enabled, it will wake up the MCU when the Watchdog Time-out period expires. If the watchdog is disabled, only an external reset or an external level triggered interrupt can wake up the MCU.

Timer / Counter

The AT90S1200 provides one general purpose 8-bit Timer/Counter. The Timer/Counter gets the prescaled clock from the 10-bit prescaling timer. The Timer/Counter can either be used as a timer with an internal clock timebase or as a counter with an external pin connection which triggers the counting.

The Timer/Counter Prescaler

Figure 19 shows the general Timer/Counter prescaler.

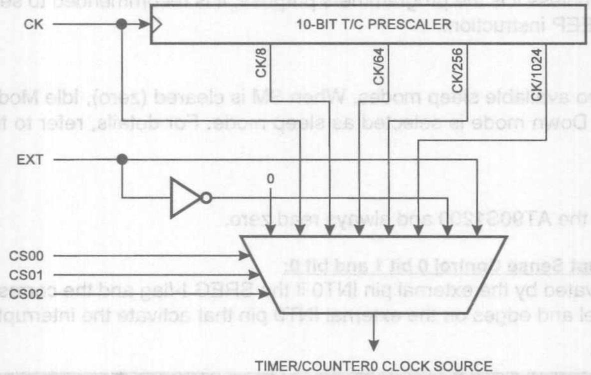


Figure 19. Timer/Counter0 Prescaler

The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. For the Timer/Counter, added selections as CK, external source and stop, can be selected as clock sources.

The 8-bit Timer/Counter0

Figure 20 shows the block diagram for Timer/Counter0.

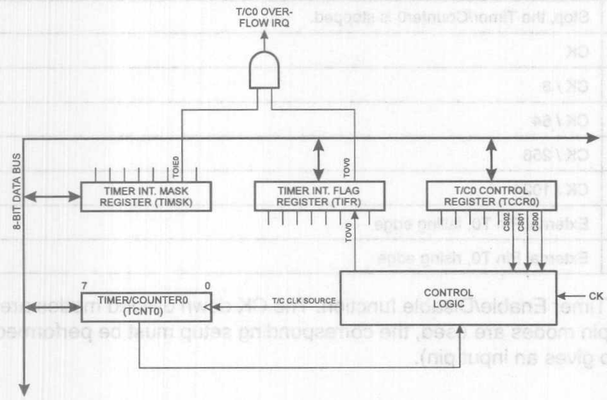


Figure 20. Timer/Counter 0 Block Diagram

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter0 Control Register - TCCR0. The overflow status flag is found in the Timer/Counter Interrupt Flag Register - TIFR. Control signals are found in the Timer/Counter0 Control Register - TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

THE TIMER/COUNTER0 CONTROL REGISTER - TCCR0

Bit	7	6	5	4	3	2	1	0	
$33	-	-	-	-	-	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..3 - Res : Reserved bits:

These bits are reserved bits in the AT90S1200 and always read zero.

Bits 2,1,0 - CS02, CS01, CS00 : Clock Select0, bit 2,1 and 0:

The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer0.

Table 5. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual data direction control register (cleared to zero gives an input pin).

Bits 5..3 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and always read zero.

THE TIMER COUNTER 0 - TCNT0

Bit	7	6	5	4	3	2	1	0	
$32	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the timer clock cycle following the write operation.

The Watchdog Timer

The Watchdog Timer is clocked from a separate on-chip oscillator which runs at 1MHz. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted from 16 to 2048 cycles. Refer to RC Oscillator Frequency graph on page 2-46. These values apply at V_{CC} = 5V. See characterization data for RC oscillator frequency at other V_{CC} voltages. The WDR - Watchdog Reset - instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the maximum period between two WDR instructions to avoid that the Watchdog Timer resets the MCU. If the reset period expires without another WDR instruction, the AT90S1200 resets and executes from the reset vector. For timing details on the Watchdog reset, refer to Page 2-17.

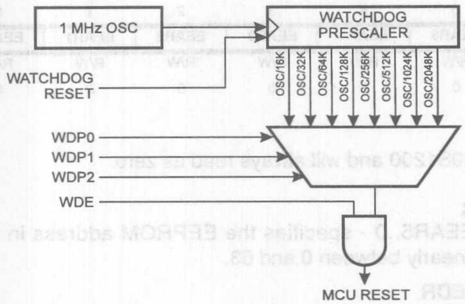


Figure 21. Watchdog Timer

THE WATCHDOG TIMER CONTROL REGISTER - WDTCR

Bit	7	6	5	4	3	2	1	0	
$21	-	-	-	-	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..4 - Res : Reserved bits:
These bits are reserved bits in the AT90S1200 and will always read as zero.

Bit 3 - WDE : Watchdog Enable:
When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled.

Bits 2..0 - WDP2..0 : Watchdog Timer Prescaler 2,1 and 0:
The WDP2..0 determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 6.

Table 6. Watchdog Timer Prescale Select (Typical Values at V_{CC} = 5.0V)

WDP2	WDP1	WDP0	Timeout Period
0	0	0	16 cycles
0	0	1	32 cycles
0	1	0	64 cycles
0	1	1	128 cycles
1	0	0	256 cycles
1	0	1	512 cycles
1	1	0	1024 cycles
1	1	1	2048 cycles

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written. An EEPROM brown-out detection prevents writing to the EEPROM if V_{CC} is below a certain level.

When the EEPROM is read or written, the CPU is halted for two clock cycles before the next instruction is executed.

THE EEPROM ADDRESS REGISTER - EEAR

Bit	7	6	5	4	3	2	1	0	
$1E	-	-	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7,6 - Res : Reserved bits:

These bits are reserved bit in the AT90S1200 and will always read as zero.

Bits 5..0 - EEAR..0 : EEPROM Address:

The EEPROM Address Register - EEAR5..0 - specifies the EEPROM address in the 64-byte EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 63.

THE EEPROM DATA REGISTER - EEDR

Bit	7	6	5	4	3	2	1	0	
$1D	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..0 - EEDR7..0 : EEPROM Data:

For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

THE EEPROM CONTROL REGISTER - EECR

Bit	7	6	5	4	3	2	1	0	
$1C	-	-	-	-	-	-	EEWE	EERE	EECR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..2 - Res : Reserved bits:

These bits are reserved bits in the AT90S1200 and will always be read as zero.

Bit 1 - EEWE : EEPROM Write Enable:

The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. When the write access time (typically 2.5ms at Vcc=5V and 4ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 - EERE : EEPROM Read Enable:

The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed.

The Analog Comparator

The analog comparator compares the input values on the positive pin PB0 (AIN0) and the negative pin PB1 (AIN1). When the voltage on the positive pin PB0 (AIN0) is higher than the voltage on the negative pin PB1 (AIN1), the Analog Comparator Output, ACO is set (one). The comparator's output can be set to trigger the Analog Comparator interrupt. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 22.

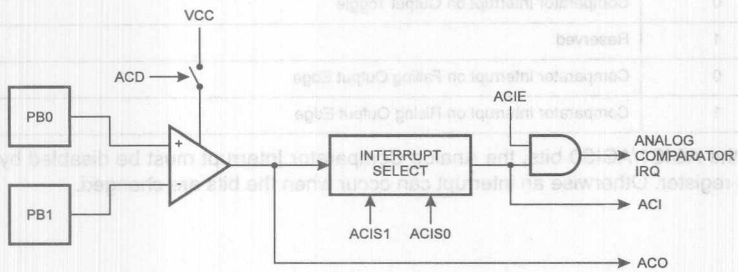


Figure 22. Analog Comparator Block Diagram

THE ANALOG COMPARATOR CONTROL AND STATUS REGISTER - ACSR

Bit	7	6	5	4	3	2	1	0	
$08	ACD	-	ACO	ACI	ACIE	-	ACIS1	ACIS0	ACSR
Read/Write	R/W	R	R	R/W	R/W	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ACD : Analog Comparator Disable

When this bit is set(one), the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. It is most commonly used if power consumption during Idle Mode is critical, and wake-up from the analog comparator is not required. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 - Res : Reserved bit:

This bit is a reserved bit in the AT90S1200 and will always read as zero.

Bit 5 - ACO : Analog Comparator Output:

ACO is directly connected to the comparator output.

Bit 4 - ACI : Analog Comparator Interrupt Flag:

This bit is set (one) when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set (one) and the I-bit in SREG is set (one). ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 - ACIE : Analog Comparator Interrupt Enable:

When the ACIE bit is set (one) and the I-bit in the Status Register is set (one), the analog comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

Bit 2 - Res : Reserved bit:

This bit is a reserved bit in the AT90S1200 and will always read as zero.

Bits 1,0 - ACIS1, ACIS0 : Analog Comparator Interrupt Mode Select:

These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 7.

Table 7. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

Note: When changing the ACIS1/ACIS0 bits, the Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

I/O-Ports

Port B

Port B is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port B, one each for the Data Register - PORTB ($18), Data Direction Register - DDRB ($17) and the Port B Input Pins - PINB ($16). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The Port B output buffers can sink 20mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The Port B pins with alternate functions are shown in the following table:

Table 8. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	AIN0 (Analog comparator positive input)
PB1	AIN1 (Analog comparator negative input)
PB5	MOSI (Data input line for memory downloading)
PB6	MISO (Data output line for memory uploading)
PB7	SCK (Serial clock input)

When the pins are used for the alternate function, the DDRB and PORTB register has to be set according to the alternate function description.

THE PORTB DATA REGISTER - PORTB

Bit	7	6	5	4	3	2	1	0	
$18	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B DATA DIRECTION REGISTER - DDRB

Bit	7	6	5	4	3	2	1	0	
$17	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B INPUT PIN ADDRESS - PINB

Bit	7	6	5	4	3	2	1	0	
$16	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port B Input Pins address - PINB - is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the PORTB Data Latch is read, and when reading PINB, the logical values present on the pins are read.

PORTB AS GENERAL DIGITAL I/O

All 8 bits in port B are equal when used as digital I/O pins.

PBn, General I/O pin: The DDBn bit in the DDRB register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) and the pin is configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, PORTBn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 9. DDBn Effect on PORTB Pins

DDBn	PORTBn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PBn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6...0, pin number.

ALTERNATE FUNCTIONS OF PORTB

The alternate pin functions of Port B are:

SCK - PORTB, Bit 7:
SCK, Clock input pin for Memory up/downloading.

MISO - PORTB, Bit 6:
MISO, Data output pin for Memory uploading.

MOSI - PORTB, Bit 5:
MOSI, Data input pin for Memory downloading.

AIN1 - PORTB, Bit 1:
AIN1, Analog Comparator Negative Input. When configured as an input (DDB1 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB1 is cleared (zero)), this pin also serves as the negative input of the on-chip analog comparator.

AIN0 - PORTB, Bit 0:
AIN0, Analog Comparator Positive Input. When configured as an input (DDB0 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB0 is cleared (zero)), this pin also serves as the positive input of the on-chip analog comparator.

PORT B SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

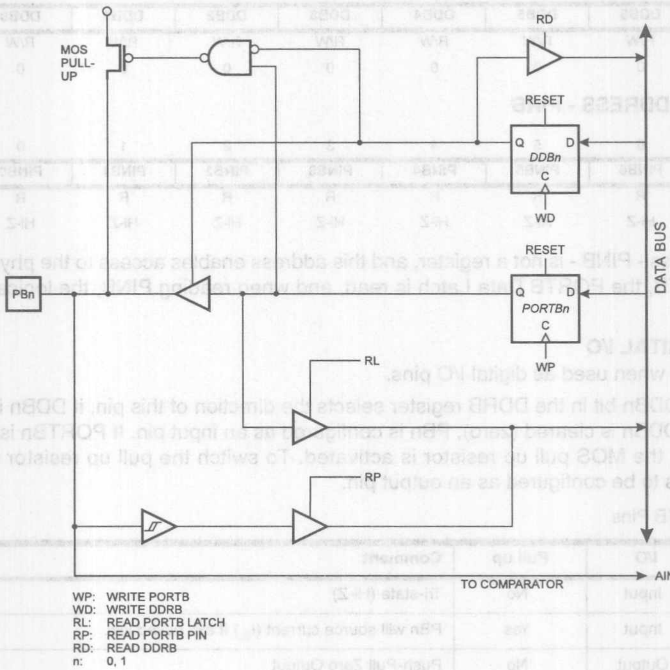


Figure 23. PORTB Schematic Diagram (pins PB0 and PB1)

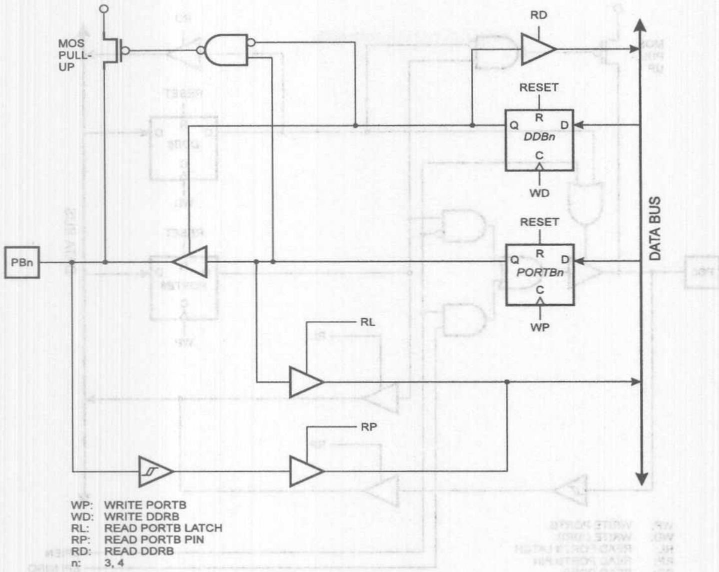


Figure 24. PORTB Schematic Diagram (Pins PB2, PB3 and PB4)

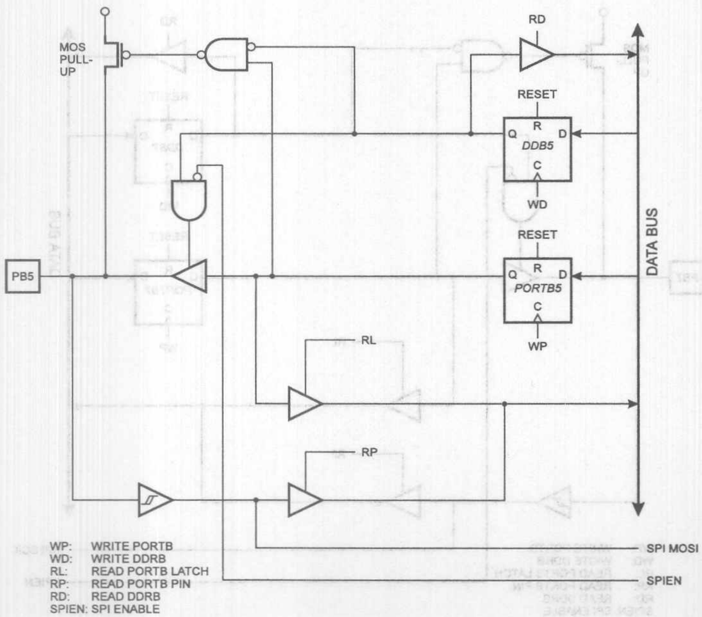


Figure 25. PORTB Schematic Diagram, Pin PB5

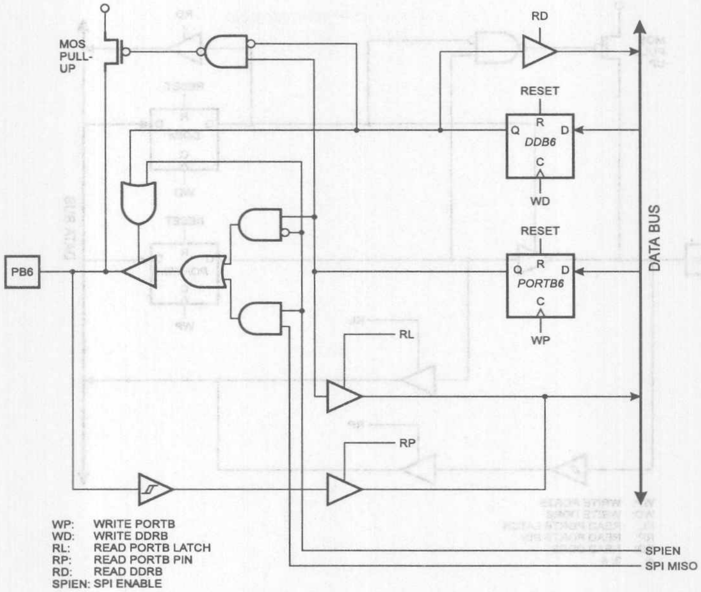


Figure 26. PORTB Schematic Diagram, Pin PB6

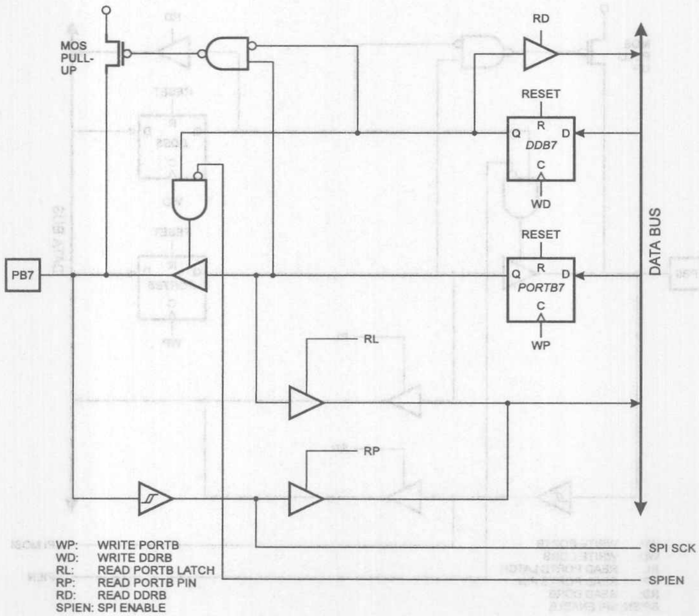


Figure 27. PORTB Schematic Diagram, Pin PB7

Port D

Three data memory address locations are allocated for the Port D, one each for the Data Register - PORTD ($12), Data Direction Register - DDRD ($11) and the Port D Input Pins - PIND ($10). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

Port D has seven bi-directional I/O pins with internal pullups, PD6..PD0. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Some Port D pins have alternate functions as shown in the following table:

Table 10. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD2	INT0 (External interrupt 0 input)
PD4	T0 (Timer/Counter 0 external input)

THE PORTD DATA REGISTER - PORTD

Bit	7	6	5	4	3	2	1	0	
$12	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D DATA DIRECTION REGISTER - DDRD

Bit	7	6	5	4	3	2	1	0	
$11	-	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D INPUT PINS ADDRESS - PIND

Bit	7	6	5	4	3	2	1	0	
$10	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port D Input Pins address - PIND - is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

PORTD AS GENERAL DIGITAL I/O

PDn, General I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PORTDn is set (one) when DDDn is configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTDn bit has to be cleared (zero) or the pin has to be configured as an output pin.

Table 11. DDDn Bits Effect on Port D Pins

DDDn	PORTDn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PDn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 6…0, pin number.

ALTERNATE FUNCTIONS FOR PORTD

The alternate functions of Port D are:

T0 - PORTD, Bit 4:

T0, Timer/Counter0 clock source. See the Timer description for further details.

INT0 - PORTD, Bit 2:

INT0, External Interrupt source 0. See the interrupt description for further details.

PORTD SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

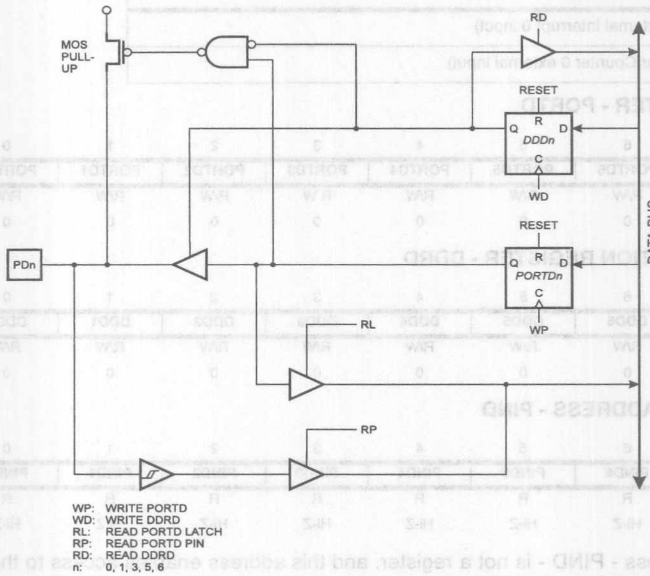


Figure 28. PORTD Schematic Diagram (Pins PD0, PD1, PD3, PD5 and PD6)

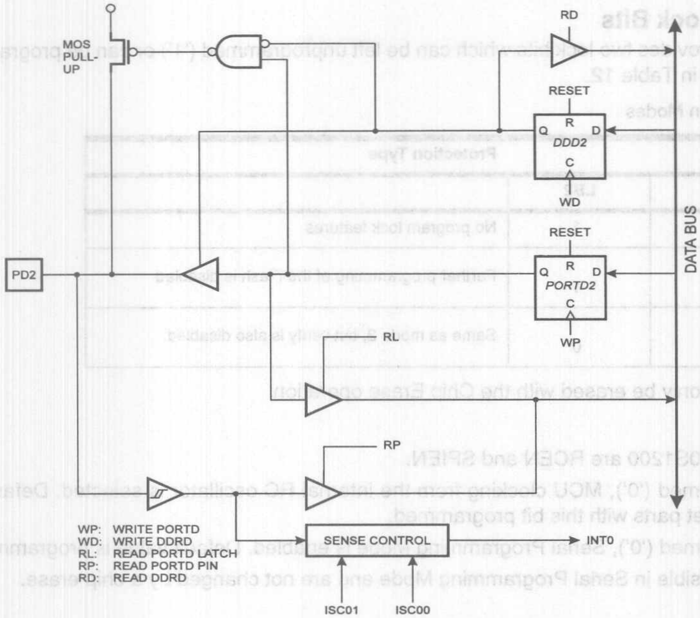


Figure 29. PORTD Schematic Diagram (Pin PD2)

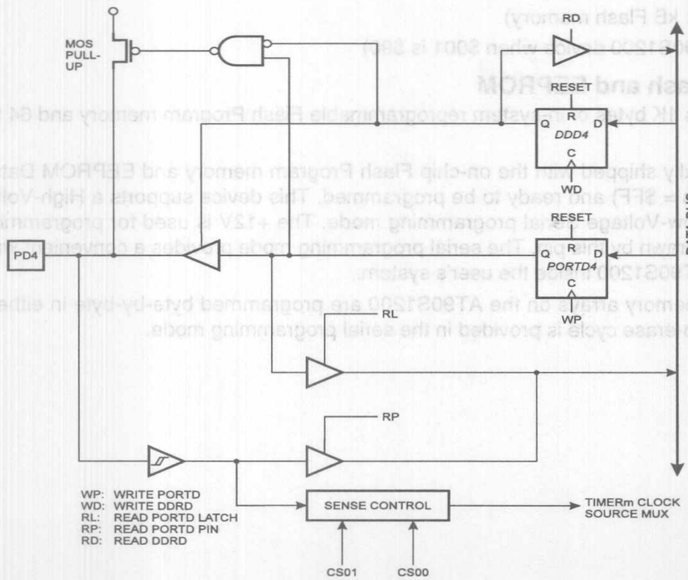


Figure 30. PORTD Schematic Diagram (Pin PD4)

Memory Programming

Program Memory Lock Bits

The AT90S1200 MCU provides two lock bits which can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table 12.

Table 12. Lock Bit Protection Modes

Program Lock Bits			Protection Type
Mode	LB1	LB2	
1	1	1	No program lock features
2	0	1	Further programming of the Flash is disabled
3	0	0	Same as mode 2, but verify is also disabled.

Note: The Lock Bits can only be erased with the Chip Erase operation.

Fuse Bits

The Fuse bits in the AT90S1200 are RCEN and SPIEN.

When RCEN is programmed ('0'), MCU clocking from the internal RC oscillator is selected. Default value is erased ('1'). Order AT90S1200A to get parts with this bit programmed.

When SPIEN is programmed ('0'), Serial Programming Mode is enabled. Default value is programmed ('0').

These bits are not accessible in Serial Programming Mode and are not changed by a chip erase.

Device Code

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode. The three bytes reside in a separate address space, and for the AT90S1200 they are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $90 (indicates 1 kB Flash memory)
3. $002: $01 (indicates 90S1200 device when $001 is $90)

Programming the Flash and EEPROM

Atmel's AT90S1200 offers 1K bytes of in-system reprogrammable Flash Program memory and 64 bytes of EEPROM Data memory.

The AT90S1200 is normally shipped with the on-chip Flash Program memory and EEPROM Data memory arrays in the erased state (i.e. contents = $FF) and ready to be programmed. This device supports a High-Voltage (12V) Parallel programming mode and a Low-Voltage Serial programming mode. The +12V is used for programming enable only, and no current of significance is drawn by this pin. The serial programming mode provides a convenient way to download the Program and Data into the AT90S1200 inside the user's system.

The Program and Data memory arrays on the AT90S1200 are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided in the serial programming mode.

Parallel Programming

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory + Program Memory Lock bits and Fuse bits in the AT90S1200.

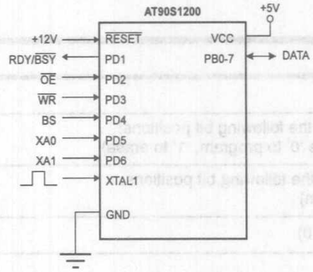


Figure 31. Parallel Programming

SIGNAL NAMES

In this section, some pins of the AT90S1200 are referenced by signal names describing their functionality during parallel programming rather than their pin names. Pins not described in the following table are referenced by pin names.

Table 13. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY/BSY	PD1	O	0: Device is busy programming, 1: Device is ready for new command
OE	PD2	I	Output Enable (Active Low)
WR	PD3	I	Write Pulse (Active Low)
BS	PD4	I	Byte Select
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1

The XA1/XA0 bits determine the action taken when the XTAL1 pin is given a positive pulse. The bit settings are shown in the following table:

Table 14. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or Low address byte for Flash determined by BS)
0	1	Load Data (High or Low data byte for Flash determined by BS)
1	0	Load Command
1	1	No Action, Idle

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action on input or output. The command is a byte where the different bits are assigned functions as shown in the following table:

Table 15. Command Byte Bit Coding

Bit#	Meaning when Set
7	Chip Erase
6	Write Fuse Bits. Located in the data byte at the following bit positions: D5: SPI Fuse, D0: RCEN Fuse (Note: write '0' to program, '1' to erase)
5	Write Lock Bits. Located in the data byte at the following bit positions: D2: LB2, D1: LB1 (Note: write '0' to program)
4	Write Flash or EEPROM (determined by bit 0)
3	Read signature row
2	Read Lock and Fuse Bits. Located in the data byte at the following bits positions: D7: LB1, D6: LB2, D5: SPI Fuse, D0: RCEN Fuse
1	Read from Flash or EEPROM (determined by bit 0)
0	0: Flash Access, 1: EEPROM Access

ENTER PROGRAMMING MODE

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5V between VCC and GND.
2. Set the $\overline{RESET}$ and BS pin to '0' and wait at least 100 ns.
3. Apply 12V to $\overline{RESET}$ and wait at least 100ns before changing BS.

CHIP ERASE

The chip erase will erase the Flash and EEPROM memories plus Lock bits. The lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. lA chip erase must be performed before the chip is programmed.

Load Command "Chip Erase"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'.
3. Set PB(7:0) to '1000 0000'. This is the command for Chip erase.
4. Give XTAL1 a positive pulse. This loads the command, and starts the erase of the Flash and EEPROM arrays. After pulsing XTAL1, give $\overline{WR}$ a negative pulse to enable lock bit erase at the end of the erase cycle. Then wait at least 10 ms for the chip erase cycle to finish. Chip erase does not generate any activity on the RDY/$\overline{BSY}$ signal.

PROGRAMMING THE FLASH

Load Command "Program Flash"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'
3. Set PB(7:0) to '0001 0000'. This is the command for Flash programming.
4. Give XTAL1 a positive pulse. This loads the command.

Load Address Low byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '0'. This selects Low address.
3. Set PB(7:0) = Address Low byte ($00 - $FF)
4. Give XTAL1 a positive pulse. This loads the Address Low byte.

Load Address High byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '1'. This selects High address.
3. Set PB(7:0) = Address High byte ($00 - $01)
4. Give XTAL1 a positive pulse. This loads the Address High byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data Low byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data Low byte.

Write Data Low byte

1. Set BS to '0'. This selects Low data.
2. Give $\overline{\text{WR}}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{\text{BSY}}$ goes low.
3. Wait until RDY/$\overline{\text{BSY}}$ goes high to program the next byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data High byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data High byte.

Write Data High byte

1. Set BS to '1'. This selects high data.
2. Give $\overline{\text{WR}}$ a positive pulse. This starts programming of the data byte. RDY/$\overline{\text{BSY}}$ goes low.
3. Wait until RDY/$\overline{\text{BSY}}$ goes high to program the next byte.

The loaded command and address are retained in the device during programming. To simplify programming, the following should be considered.

- The command for Flash programming needs only be loaded before programming of the first byte.
- Address High byte needs only be loaded before programming a new 256 word page in the Flash.

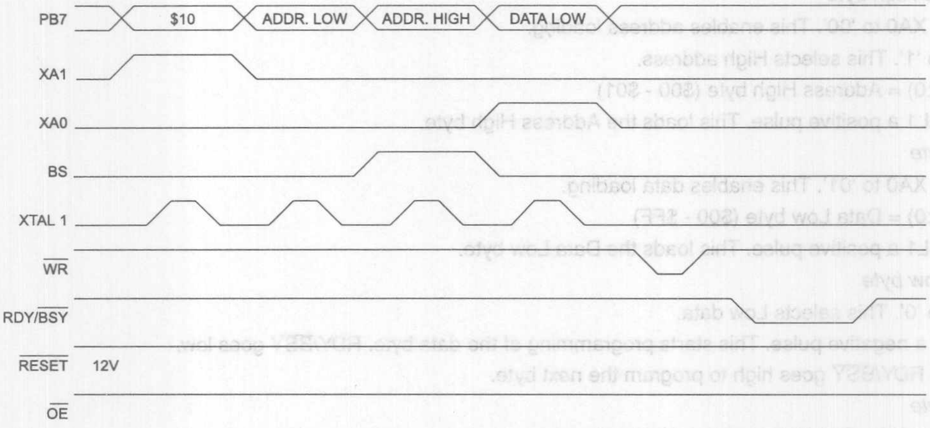


Figure 32. Programming Flash Low Byte

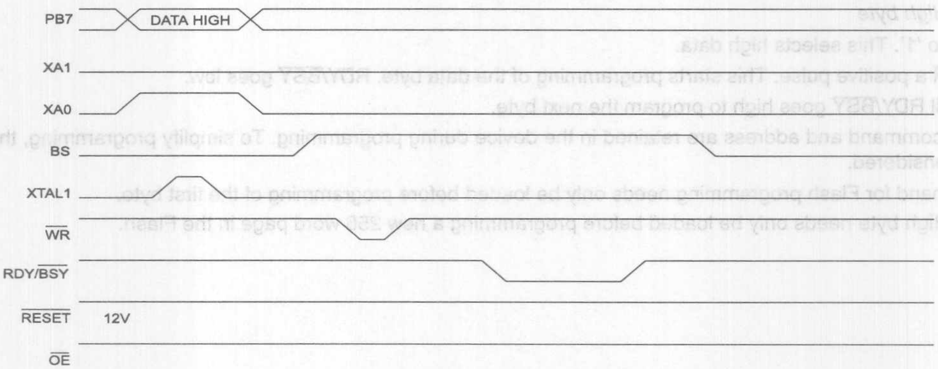


Figure 33. Programming Flash High Byte

PROGRAMMING THE EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0001 0001'.
2. Load Low EEPROM Address ($00 - $3F)
3. Load Low EEPROM Data ($00 - $FF)
4. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

The Command needs only be loaded before programming the first byte.

READING THE FLASH

The algorithm for reading the Flash memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0010'.
2. Load Low Address ($00 - $FF)
3. Load High Address ($00 - $01)
4. Set $\overline{\text{OE}}$ to '0', and BS to '0'. The Low Data byte can now be read at PB(7:0)
5. Set BS to '1'. The High Data byte can now be read from PB(7:0)
6. Set $\overline{\text{OE}}$ to '1'.

The Command needs only be loaded before reading the first byte.

READING THE EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0011'.
2. Load Low Address ($00 - $3F)
3. Set $\overline{\text{OE}}$ to '0', and BS to '0'. The EEPROM Data byte can now be read at PB(7:0)
4. Set $\overline{\text{OE}}$ to '1'.

The Command needs only be loaded before reading the first byte.

PROGRAMMING THE FUSE BITS

The algorithm for programming the Fuse bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0100 0000'.
2. Load Data.

   Bit 5 = '0' programs the SPI Fuse bit. Bit 5 = '1' erases the SPI Fuse bit.

   Bit 0 = '0' programs the RCEN Fuse bit. Bit 0 = '1' erases the RCEN Fuse bit.
3. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

IMPORTANT! $\overline{\text{WR}}$ *must be held low for at least 1 ms.*

PROGRAMMING THE LOCK BITS

The algorithm for programming the Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0010 0000'.
2. Load Data.

   Bit 2 = '0' programs Lock Bit2

   Bit 1 = '0' programs Lock Bit1
3. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

The lock bits can only be cleared by executing a chip erase.

READING THE FUSE AND LOCK BITS

The algorithm for reading the Fuse and Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0100'.
2. Set $\overline{OE}$ to '0', and BS to '1'. The Status of Fuses an Lock bits can now be read at PB(7:0)
   Bit 7: Lock Bit1 ('0' means programmed)
   Bit 6: Lock Bit2 ('0' means programmed)
   Bit 5: SPI Fuse ('0' means programmed)
   Bit 0: RCEN Fuse ('0' means programmed)
3. Set $\overline{OE}$ to '1'.

Observe especially that BS needs to be set to '1'.

READING THE SIGNATURE BYTES

The algorithm for reading the Signature bytes is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 1000'.
2. Load Low address ($00 - $02)
   Set $\overline{OE}$ to '0', and BS to '0'. The Selected Signature byte can now be read at PB(7:0)
3. Set $\overline{OE}$ to '1'.

The command needs only be programmed before reading the first byte.

Serial Downloading

Both the Program and Data memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into $FF.

The Program and EEPROM memory arrays have separate address spaces: $000 to $3FF for Program Flash memory and $000 to $03F for EEPROM Data memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 1 XTAL1 clock cycle

High: > 4 XTAL1 clock cycles

SERIAL PROGRAMMING ALGORITHM

To program and verify the AT90S1200 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 16):

1. *Power-up sequence*:
   Apply power between VCC and GND while $\overline{RESET}$ and SCK are set to '0'. (If the programmer can not guarantee that SCK is held low during power-up, $\overline{RESET}$ must be given a positive pulse after SCK has been set to '0'.) If a crystal is not connected across pins XTAL1 and XTAL2, apply a 0 to 16 MHz clock to the XTAL1 pin.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB5. Refer to the above section for minimum low and high periods for the serial clock input, SCK.
3. If a chip erase is performed (must be done to erase the Flash), wait 10ms, give $\overline{RESET}$ a positive pulse and start over again from Step 2.
4. The Flash or EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. The next byte can be written after 4 ms.

5. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.

At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.

6. Power-off sequence (if needed):
   Set XTAL1 to '0' (if a crystal is not used).
   Set $\overline{\text{RESET}}$ to '1'.
   Turn V_{CC} power off.

Table 16. Serial Programming Instruction Set AT90S1200

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after $\overline{\text{RESET}}$ goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both 1K & 64 byte memory arrays
Read Program Memory	0010 **H**000	0000 000**a**	**bbbb bbbb**	**oooo oooo**	Read **H**(high or low) data **o** from Program memory at word address **a:b**
Write Program Memory	0100 **H**000	0000 000**a**	**bbbb bbbb**	**iiii iiii**	Write **H**(high or low) data **i** to Program memory at word address **a:b**
Read EEPROM Memory	1010 0000	0000 0000	xx**bb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**
Write EEPROM Memory	1100 0000	0000 0000	xx**bb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**
Write Lock Bits	1010 1100	111x x***21***x	xxxx xxxx	xxxx xxxx	Write lock bits. Set bits ***1,2***='0' to program lock bits.
Read Device Code	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Device Code **o** from address **b**.

Notes: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1- High byte
o = data out
i = data in
x = don't care
1 = lock bit 1
2 = lock bit 2

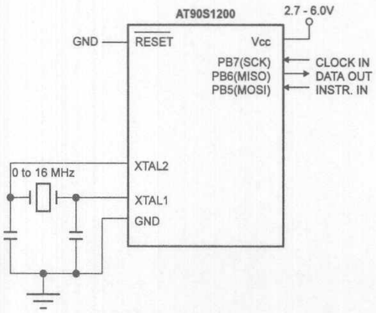


Figure 34. Programming and Verify

When *writing* serial data to the AT90S1200, data is clocked on the *rising* edge of CLK.

When *reading* data from the AT90S1200, data is clocked on the *falling* edge of CLK. See Figure 35 for an explanation.

Programming Characteristics

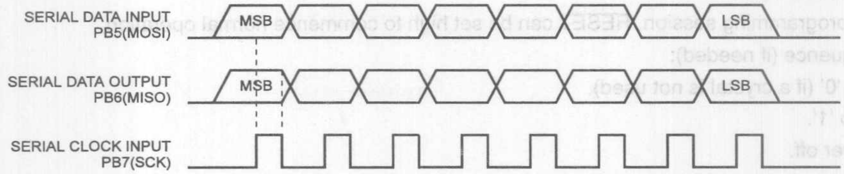


Figure 35. Serial Downloading Waveforms

Absolute Maximum Ratings

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Current per I/O Pin	40.0 mA
DC Current VCC and GND Pins	140.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

2

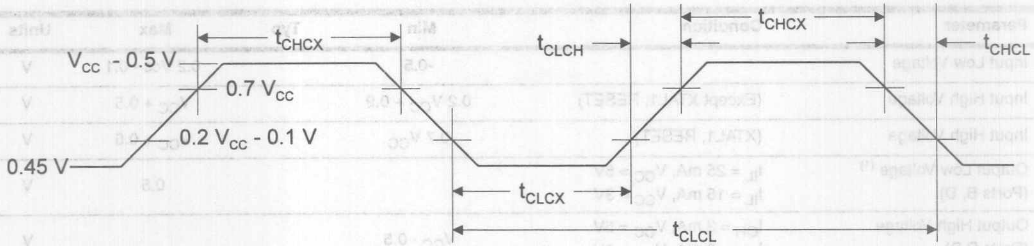
DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		0.2 Vcc - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, $\overline{RESET}$)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, $\overline{RESET}$)	0.7 V_{CC}		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage [1] (Ports B, D)	I_{IL} = 25 mA, V_{CC} = 5V I_{IL} = 15 mA, V_{CC} = 3V			0.5	V
V_{OH}	Output High Voltage (Ports B,D)	I_{OH} = 3 mA, V_{CC} = 5V I_{OH} = 3 mA, V_{CC} = 3V	V_{CC} - 0.5			V
I_{OH}	Output Source Current (Ports B,D)	V_{CC} = 5V, V_{OH} = 4.5V V_{CC} = 3V, V_{OH} = 2.7V		4 2		mA
I_{IL}	Output Sink Current (Port B,D)	V_{CC} = 5V, V_{OL} = 0.5V V_{CC} = 3V, V_{OL} = 0.3V		28 11		mA
RRST	Reset Pull-Up Resistor		100		500	kΩ
$R_{I/O}$	I/O Pin Pull-Up Resistor		35		120	kΩ
I_{CC}	Power Supply Current	Active Mode, 3V, 4MHz		2		mA
		Idle Mode 3V, 4MHz		500		µA
		Power Down [2] WDT enabled, 3V		10	15 [3]	µA
		Power Down [2] WDT disabled, 3V		0.15	1[3]	µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V			20	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{IN} = 1V		10		nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 20mA
   Maximum total I_{OL} for all output pins: 80mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification.
   Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power Down is 2V.
3. Value tested to 45°C

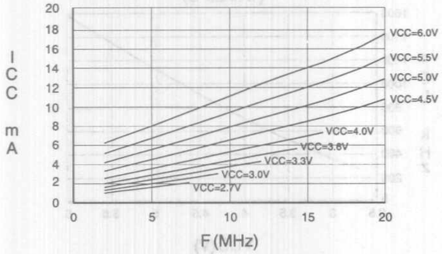
External Clock Drive Waveforms



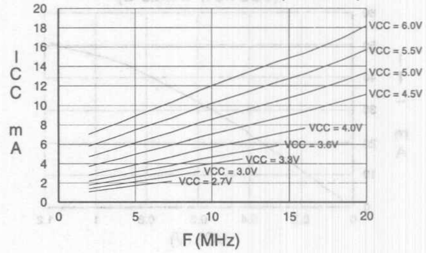
External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	16	MHz
t_{CLCL}	Clock Period	250		62.5		ns
t_{CHCX}	High Time	40		16.7		ns
t_{CLCX}	Low Time	40		16.7		ns
t_{CLCH}	Rise Time		10		4.15	ns
t_{CHCL}	Fall Time		10		4.15	ns

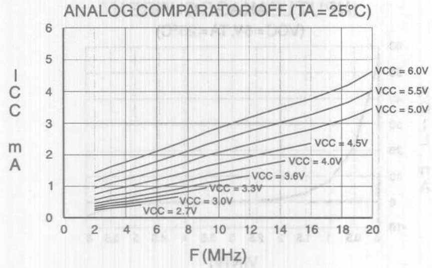
ACTIVE SUPPLY CURRENT
ANALOG COMPARATOR OFF (TA = 25°C)



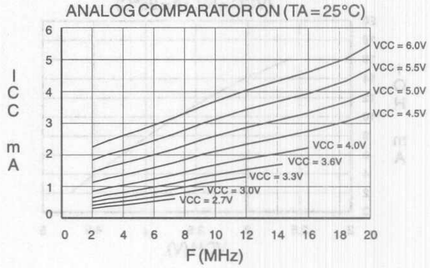
ACTIVE SUPPLY CURRENT
ANALOG COMPARATOR ON (TA = 25°C)



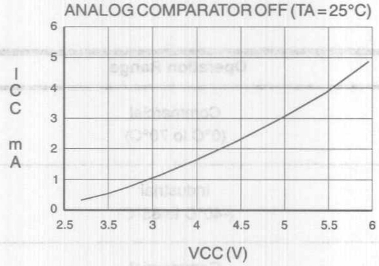
IDLE SUPPLY CURRENT
ANALOG COMPARATOR OFF (TA = 25°C)



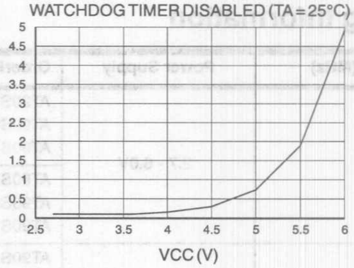
IDLE SUPPLY CURRENT
ANALOG COMPARATOR ON (TA = 25°C)



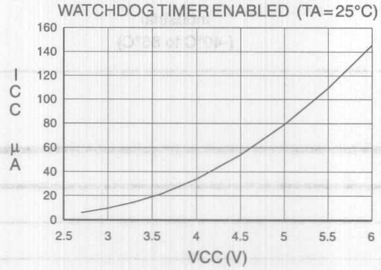
ACTIVE CURRENT RC-OSC
ANALOG COMPARATOR OFF (TA = 25°C)



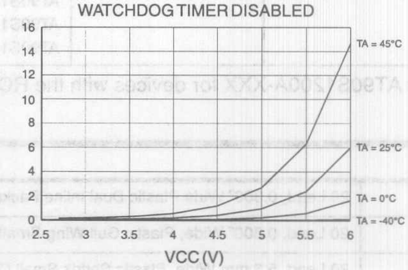
POWER DOWN SUPPLY CURRENT
WATCHDOG TIMER DISABLED (TA = 25°C)



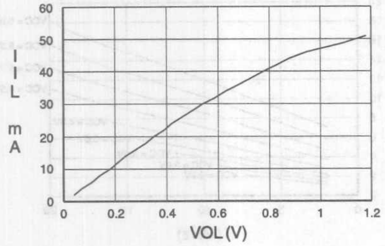
POWER DOWN SUPPLY CURRENT
WATCHDOG TIMER ENABLED (TA = 25°C)



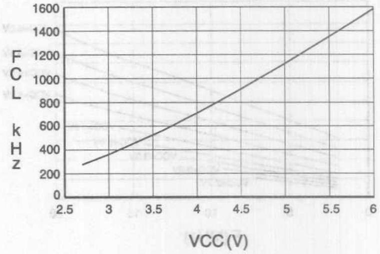
POWER DOWN SUPPLY CURRENT
WATCHDOG TIMER DISABLED



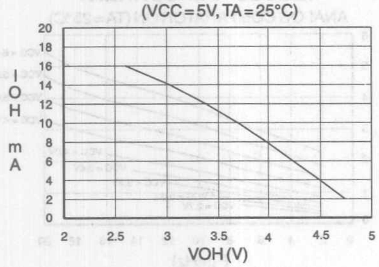
INPUT SINK CURRENT
(VCC=5V, TA=25°C)



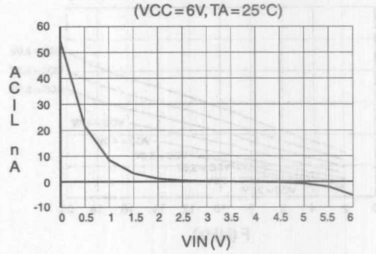
RC OSCILLATOR FREQUENCY
(TA=25°C)



OUTPUT SOURCE CURRENT
(VCC=5V, TA=25°C)



ANALOG COMPARATOR
INPUT LEAKAGE CURRENT
(VCC=6V, TA=25°C)



Note: Charts show typical values.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code*	Package	Operation Range
4	2.7 - 6.0V	AT90S1200-4PC AT90S1200-4SC AT90S1200-4YC	20P3 20S 20Y	Commercial (0°C to 70°C)
		AT90S1200-4PI AT90S1200-4SI AT90S1200-4YI	20P3 20S 20Y	Industrial (-40°C to 85°C)
16	4.0 - 6.0V	AT90S1200-16PC AT90S1200-16SC AT90S1200-16YC	20P3 20S 20Y	Commercial (0°C to 70°C)
		AT90S1200-16PI AT90S1200-16SI AT90S1200-16YI	20P3 20S 20Y	Industrial (-40°C to 85°C)

* Order AT90S1200A-XXX for devices with the RCEN fuse programmed.

Package Type	
20P3	20 Lead, 0.300" Wide Plastic Dual Inline Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)
20Y	20 Lead, 5.3 mm Wide, Plastic Shrink Small Outline Package (SSOP)

AT90S1200 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F	SREG	I	T	H	S	V	N	Z	C	2-12
$3E	Reserved									
$3D	Reserved									
$3C	Reserved									
$3B	GIMSK	-	INT0	-	-	-	-	-	-	2-17
$3A	Reserved									
$39	TIMSK	-	-	-	-	-	-	TOIE0	-	2-18
$38	TIFR	-	-	-	-	-	-	TOV0	-	2-18
$37	Reserved									
$36	Reserved									
$35	MCUCR	-	-	SE	SM	-	-	ISC01	ISC00	2-19
$34	Reserved									
$33	TCCR0	-	-	-	-	-	CS02	CS01	CS00	2-21
$32	TCNT0	Timer/Counter0 (8 Bit)								2-22
$31	Reserved									
$30	Reserved									
$2F	Reserved									
$2E	Reserved									
$2D	Reserved									
$2C	Reserved									
$2B	Reserved									
$2A	Reserved									
$29	Reserved									
$28	Reserved									
$27	Reserved									
$26	Reserved									
$25	Reserved									
$24	Reserved									
$23	Reserved									
$22	Reserved									
$21	WDTCR	-	-	-	-	WDE	WDP2	WDP1	WDP0	2-23
$20	Reserved									
$1F	Reserved									
$1E	EEAR	-	EEPROM Address Register							2-24
$1D	EEDR	EEPROM Data Register								2-24
$1C	EECR	-	-	-	-	-	-	EEWE	EERE	2-24
$1B	Reserved									
$1A	Reserved									
$19	Reserved									
$18	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	2-26
$17	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	2-27
$16	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	2-27
$15	Reserved									
$14	Reserved									
$13	Reserved									
$12	PORTD	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	2-31
$11	DDRD	-	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	2-31
$10	PIND	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	2-31
$0F	Reserved									
$0E	Reserved									
$0D	Reserved									
$0C	Reserved									
$0B	Reserved									
$0A	Reserved									
$09	Reserved									
$08	ACSR	ACD	-	ACO	ACI	ACIE	-	ACIS1	ACIS0	2-25
…	Reserved									
$00	Reserved									

AT90S1200 Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd⊕Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (FFh - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd⊕Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2
CP	Rd,Rr	Compare	Rd - Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd - K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

(continued)

AT90S1200 Instruction Set Summary (Continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TRANSFER INSTRUCTIONS					
LD	Rd,Z	Load Register Indirect	Rd ← (Z)	None	2
ST	Z,Rr	Store Register Indirect	(Z) ← Rr	None	2
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	3
WDR		Watch Dog Reset	(see specific descr. for WDR/timer)	None	1

Contents

8-Bit AVR® Microcontroller with 2K bytes Downloadable Flash

Preliminary

Features

- Utilizes the *AVR*® Enhanced RISC Architecture
- AV*R* - High Performance and Low Power RISC Architecture
- 120 Powerful Instructions - Most Single Clock Cycle Execution
- 2K bytes of In-System Reprogrammable Downloadable Flash
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 128 bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 128 bytes Internal RAM
- 32 x 8 General Purpose Working Registers
- 15 Programmable I/O Lines
- V_{CC}: 2.7 - 6.0V
- Fully Static Operation, 0 - 20 MHz
- Instruction Cycle Time: 50 ns @ 20 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- One 16-Bit Timer/Counter with Separate Prescaler and Compare and Capture Modes
- Full Duplex UART
- Selectable 8, 9 or 10 bit PWM
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes
- Programming lock for Software Security
- 20-Pin Device

Description

The AT90S2313 is a low-power CMOS 8-bit microcontroller based on the *AVR*® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S2313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

(continued)

Pin Configuration

PDIP/SOIC

RESET	1	20	VCC
(RXD) PD0	2	19	PB7 (SCK)
(TXD) PD1	3	18	PB6 (MISO)
XTAL2	4	17	PB5 (MOSI)
XTAL1	5	16	PB4
(INT0) PD2	6	15	PB3 (OC1)
(INT1) PD3	7	14	PB2
(T0) PD4	8	13	PB1 (AIN1)
(T1) PD5	9	12	PB0 (AIN0)
GND	10	11	PD6 (ICP)

0839A

Block Diagram

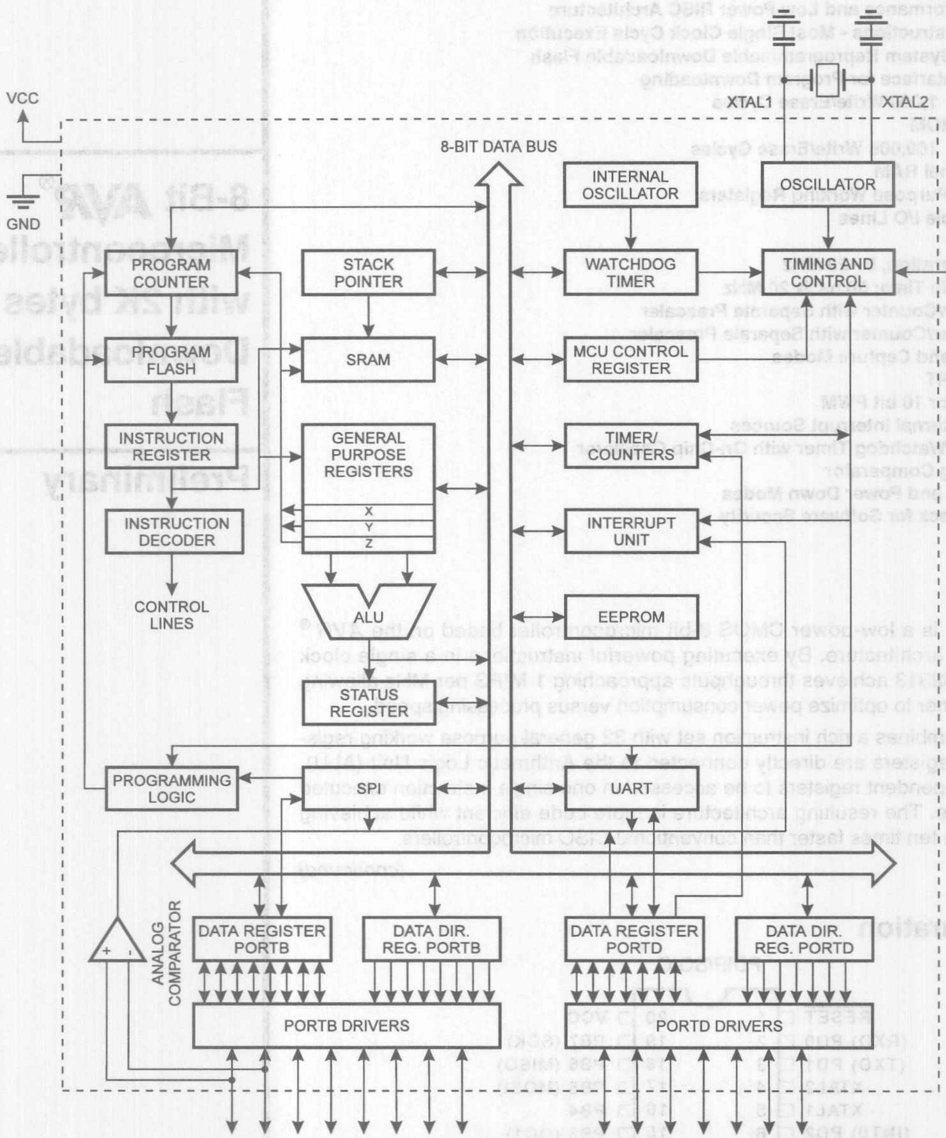


Figure 1. The AT90S2313 Block Diagram

Description (Continued)

The AT90S2313 provides the following features: 2K bytes of Downloadable Flash, 128 bytes EEPROM, 128 bytes SRAM, 15 general purpose I/O lines, 32 general purpose working registers, flexible timer/counters with compare modes, internal and external interrupts, a programmable serial UART, programmable Watchdog Timer with internal oscillator, an SPI serial port for Flash Memory downloading and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip Downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining an enhanced RISC 8-bit CPU with Downloadable Flash on a monolithic chip, the Atmel AT90S2313 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90S2313 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Supply voltage pin.

GND

Ground pin.

Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port. Port pins can provide internal pullups (selected for each bit). PB0 and PB1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the on-chip analog comparator. The Port B output buffers can sink 20mA and can drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

Port B also serves the functions of various special features of the AT90S2313 as listed on Page 3-50.

Port D (PD6..PD0)

Port D has seven bi-directional I/O pins with internal pullups, PD6..PD0. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port D also serves the functions of various special features of the AT90S2313 as listed on Page 3-55.

$\overline{\text{RESET}}$

Reset input. A low on this pin for two machine cycles while the oscillator is running resets the device.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively , of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

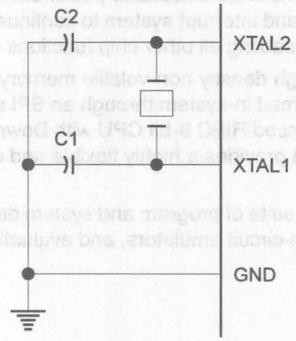


Figure 2. Oscillator Connections

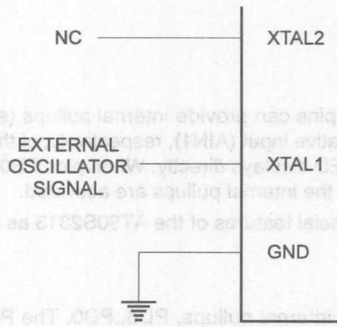


Figure 3. External Clock Drive Configuration

AT90S2313 *AVR* Enhanced RISC Microcontroller CPU

The AT90S2313 AVR RISC microcontroller is upward compatible with the AVR Enhanced RISC Architecture. The programs written for the AT90S2313 MCU are fully compatible with the range of AVR 8-bit MCUs (AT90Sxxxx) with respect to source code and clock cycles for execution.

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one ALU (Arithmetic Logic Unit) operation is executed. Two operands are output from the register file, the operation is executed, and the result is stored back in the register file - in one clock cycle.

Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing - enabling efficient address calculations. One of the three address pointers is also used as the address pointer for the constant table look up function. These added function registers are the 16-bits X-register, Y-register and Z-register.

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S2313 *AVR* Enhanced RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, and other I/O functions. The I/O memory can be accessed directly, or as the Data Space locations following those of the register file, $20 - $5F.

The *AVR* has Harvard architecture - with separate memories and buses for program and data. The program memory is accessed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system downloadable Flash memory.

With the relative jump and call instructions, the whole 1K address space is directly accessed. Most *AVR* instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 8-bit stack pointer SP is read/write accessible in the I/O space.

The 128 bytes data SRAM + register file and I/O registers can be easily accessed through the five different addressing modes supported in the *AVR* architecture.

The memory spaces in the *AVR* architecture are all linear and regular memory maps.

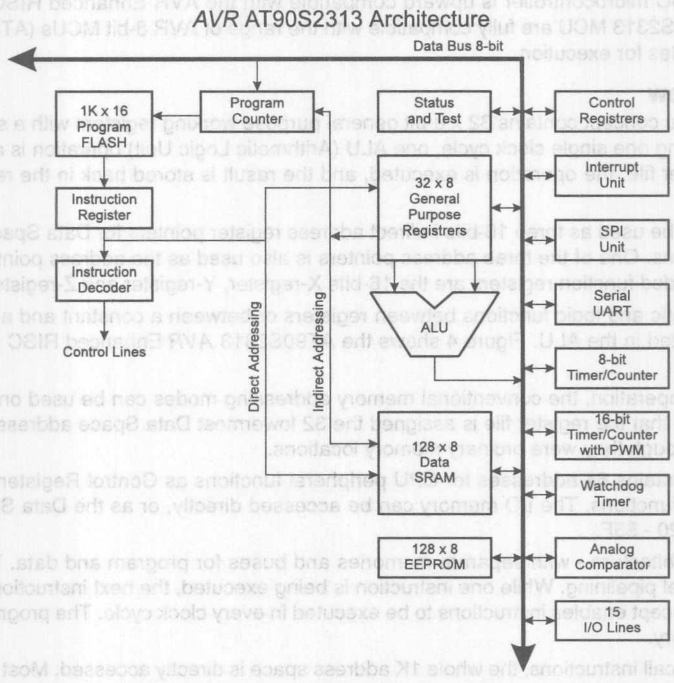


Figure 4. The AT90S2313 AVR Enhanced RISC Architecture

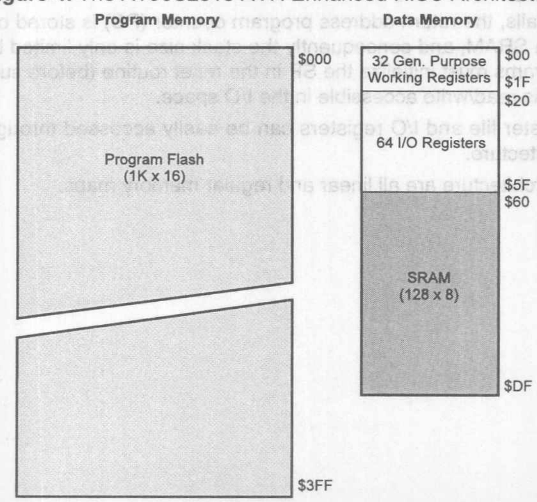


Figure 5. Memory Maps

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt address vector the higher priority.

The General Purpose Register File

Figure 6 shows the structure of the 32 general purpose registers in the CPU.

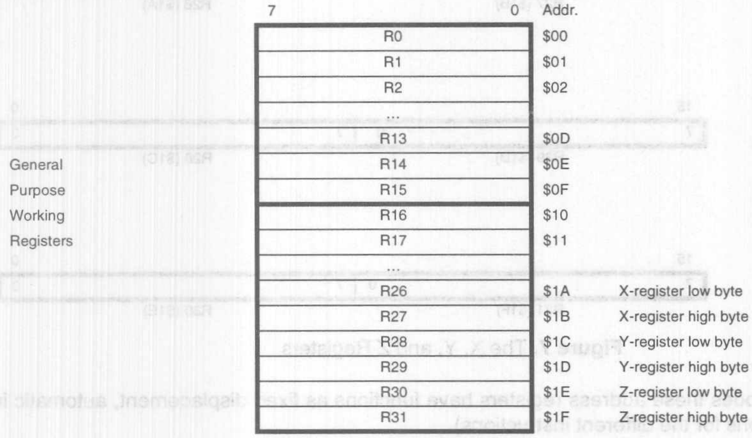


Figure 6. *AVR* CPU General Purpose Working Registers

All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file - R16..R31. The general SBC, SUB, CP, AND, OR and all other operations between two registers or on a single register apply to the entire register file.

As shown in Figure 6, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although the register file is not physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X , Y and Z registers can be set to index any register in the file.

THE X-REGISTER, Y-REGISTER, AND Z-REGISTER

The registers R26..R31 have some added functions to their general purpose usage. These registers are the address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y and Z are defined as:

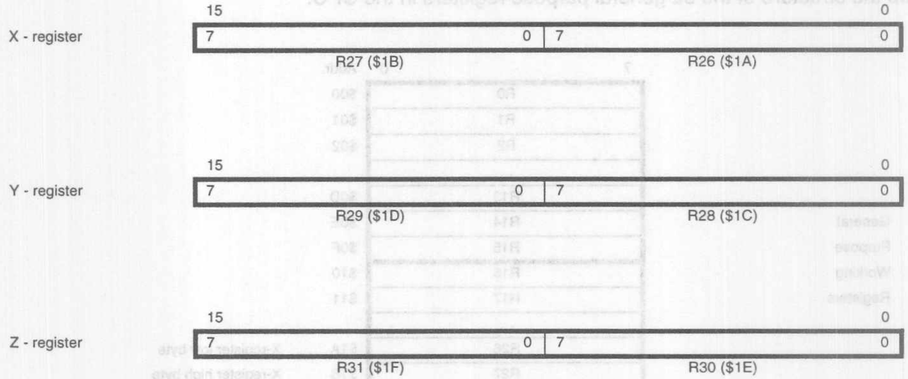


Figure 7. The X, Y, and Z Registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

The ALU - Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories - arithmetic, logic and bit-functions. Some microcontrollers in the AVR product family feature a hardware multiplier in the arithmetic part of the ALU.

The Downloadable Flash Program Memory

The AT90S2313 contains 2K bytes on-chip downloadable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 1K x 16. The Flash memory has an endurance of at least 1000 write/erase cycles.

The AT90S2313 Program Counter PC is 10 bits wide, thus addressing the 1024 program memory addresses.

See Page 3-59 for a detailed description on Flash data downloading.

Constant tables must be allocated within the address 0-2K (see the LPM - Load Program Memory instruction description).

See Page 3-14 for the different addressing modes.

The EEPROM Data Memory

The AT90S2313 contains 128 bytes of EEPROM data memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on Page 3-40 specifying the EEPROM address register, the EEPROM data register, and the EEPROM control register.

For the SPI data downloading, see Page 3-64 for a detailed description.

The SRAM Data Memory

The following figure shows how the AT90S2313 Data Memory is organized:

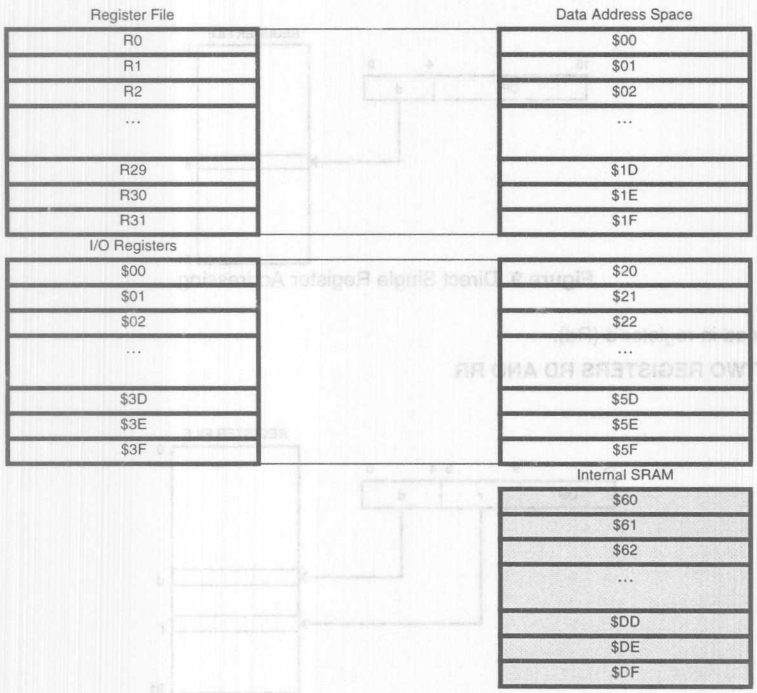


Figure 8. SRAM Organization

The 224 Data Memory locations address the Register file, I/O Memory and the data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 128 locations address the data SRAM.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-Decrement and Indirect with Post-Increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The Direct addressing reaches the entire data address space.

The Indirect with Displacement mode features 63 address locations reach from the base address given by the Y and Z register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y and Z are used and decremented and incremented.

The 32 general purpose working registers, 64 I/O registers and the 128 bytes of data SRAM in the AT90S2313 are all directly accessible through all these addressing modes.

The Program and Data Addressing Modes

The AT90S2313 AVR Enhanced RISC Microcontroller supports powerful and efficient addressing modes for access to the program memory (Flash) and data memory. This section describes the different addressing modes supported by the AVR architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

REGISTER DIRECT, SINGLE REGISTER RD

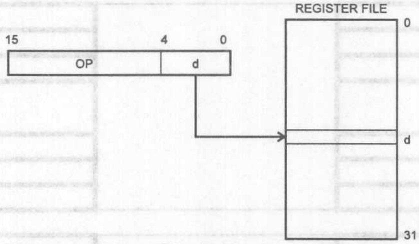


Figure 9. Direct Single Register Addressing

The operand is contained in register d (Rd).

REGISTER DIRECT, TWO REGISTERS RD AND RR

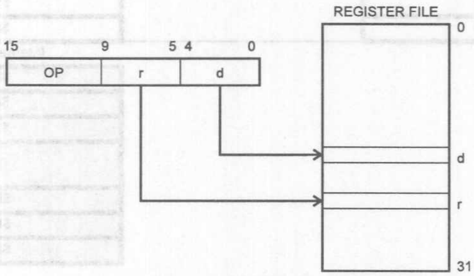


Figure 10. Direct Register Addressing, Two Registers

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O DIRECT

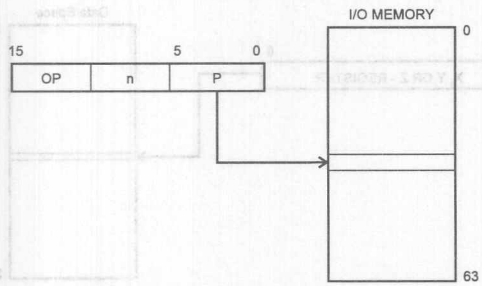


Figure 11. I/O Direct Addressing

Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

DATA DIRECT

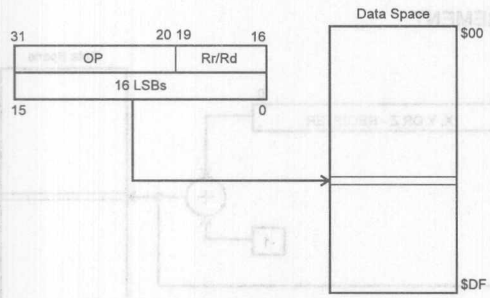


Figure 12. Direct Data Addressing

A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

DATA INDIRECT WITH DISPLACEMENT

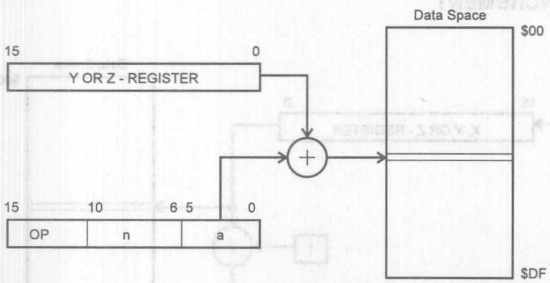


Figure 13. Data Indirect with Displacement

Operand address is the result of the Y or Z-register contents added to the address contained in 6 bits of the instruction word.

DATA INDIRECT

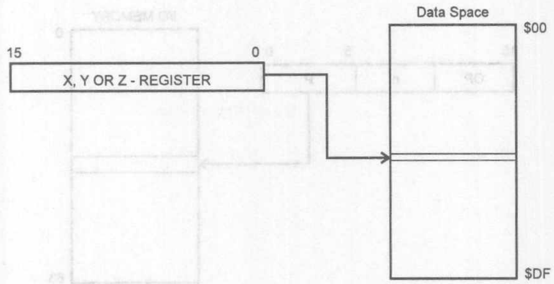


Figure 14. Data Indirect Addressing

Operand address is the contents of the X, Y or the Z-register.

DATA INDIRECT WITH PRE-DECREMENT

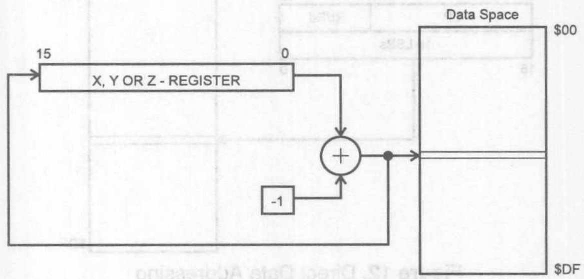


Figure 15. Data Indirect Addressing With Pre-Decrement

The X , Y or the Z-register is decremented before the operation. Operand address is the decremented contents of the X, Y or the Z-register.

DATA INDIRECT WITH POST-INCREMENT

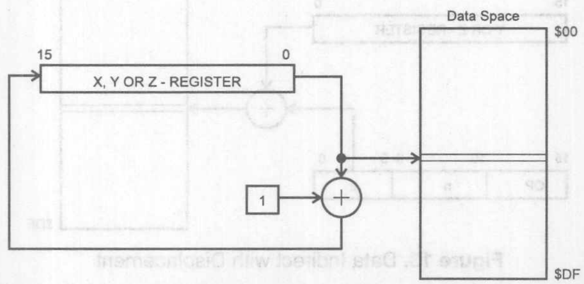


Figure 16. Data Indirect Addressing With Post-Increment

The X, Y or the Z-register is incremented after the operation. Operand address is the content of the X, Y or the Z-register prior to incrementing.

CONSTANT ADDRESSING USING THE LPM INSTRUCTION

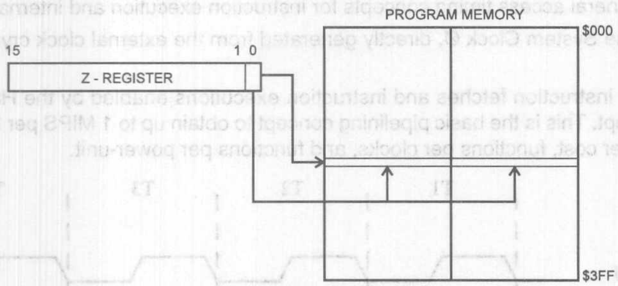


Figure 17. Code Memory Constant Addressing

Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 1K), and LSB selects low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

INDIRECT PROGRAM ADDRESSING, IJMP AND ICALL

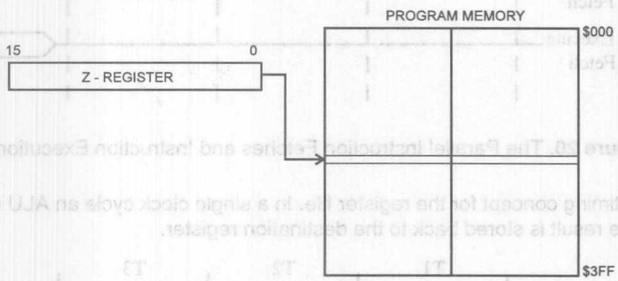


Figure 18. Indirect Program Memory Addressing

Program execution continues at address contained by the Z-register (i.e., the PC is loaded with the content of the Z-register).

RELATIVE PROGRAM ADDRESSING, RJMP AND RCALL

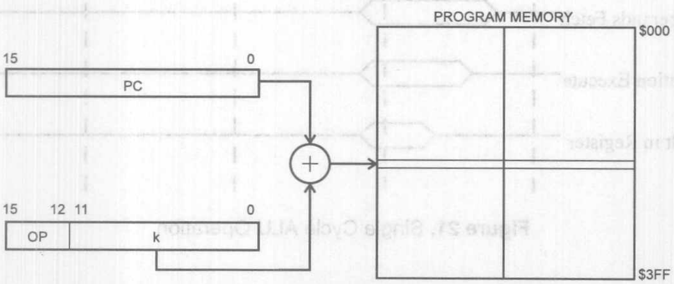


Figure 19. Relative Program Memory Addressing

Program execution continues at address PC + k. The relative address k is in the range from -2K to +(2K - 1).

Memory Access and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The *AVR* CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 20 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

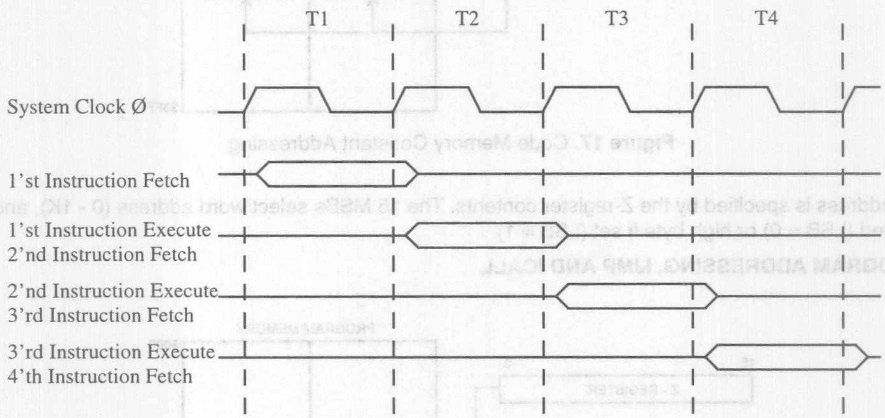


Figure 20. The Parallel Instruction Fetches and Instruction Executions

Figure 21 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

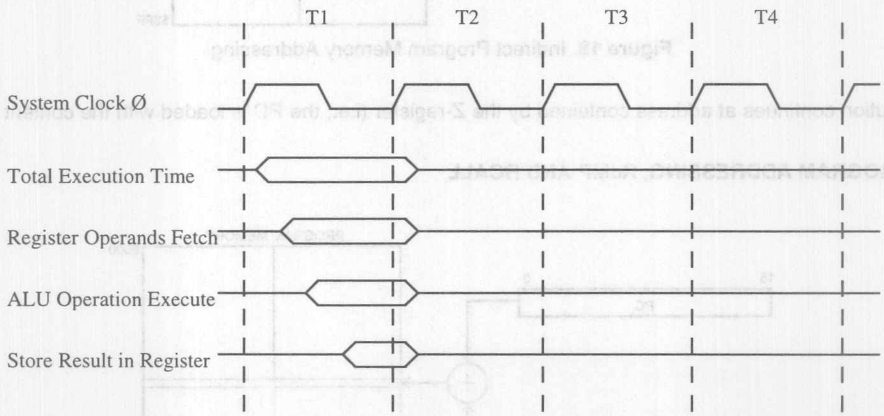


Figure 21. Single Cycle ALU Operation

The internal data SRAM access is performed in two System Clock cycles as described in Figure 22.

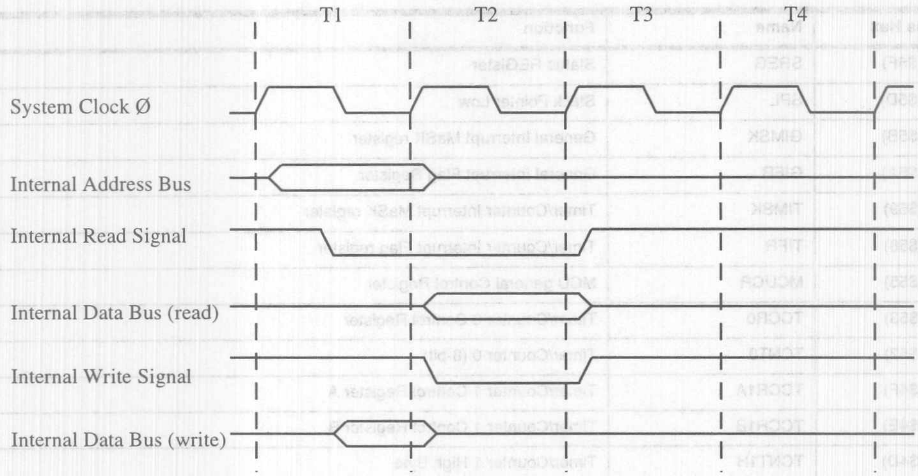


Figure 22. On-Chip Data SRAM Access Cycles

I/O Memory

The I/O space definition of the AT90S2313 is shown in the following table:

Table 1. AT90S2313 I/O Space

Address Hex	Name	Function
$3F ($5F)	SREG	Status REGister
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($5A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU general Control Register
$33 ($53)	TCCR0	Timer/Counter 0 Control Register
$32 ($52)	TCNT0	Timer/Counter 0 (8-bit)
$2F ($4F)	TCCR1A	Timer/Counter 1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter 1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter 1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter 1 Low Byte
$2B ($4B)	OCR1H	Output Compare Register 1 High Byte
$2A ($4A)	OCR1L	Output Compare Register 1 Low Byte
$25 ($45)	ICR1H	T/C 1 Input Capture Register High Byte
$24 ($44)	ICR1L	T/C 1 Input Capture Register Low Byte
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1E ($3E)	EEAR	EEPROM Address Register
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	USR	UART Status Register
$0A ($2A)	UCR	UART Control Register
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register

Note: Reserved and unused locations are not shown in the table.

All the different AT90S2313 I/O and peripherals are placed in the I/O space. The different I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range \$00 - \$1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set chapter for more details.

When using the I/O specific commands, IN, OUT,SBIS and SBIC, the I/O addresses \$00 - \$3F must be used. When addressing I/O registers as SRAM, \$20 must be added to this address. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

The different I/O and peripherals control registers are explained in the following sections.

THE STATUS REGISTER - SREG

The AVR status register - SREG - at I/O space location \$3F (\$5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
\$3F (\$5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - I : Global Interrupt Enable:

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in the interrupt mask registers - GIMSK and TIMSK. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled independent of the GIMSK and TIMSK values. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

Bit 6 - T : Bit Copy Storage:

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

Bit 5 - H : Half Carry Flag:

The half carry flag H indicates a half carry in some arithmetic operations. See the Instruction Set Description for detailed information.

Bit 4 - S : Sign Bit, $S = N \oplus V$:

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set Description for detailed information.

Bit 3 - V : Two's Complement Overflow Flag:

The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set Description for detailed information.

Bit 2 - N : Negative Flag:

The negative flag N indicates a negative result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 1 - Z : Zero Flag:

The zero flag Z indicates a zero result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 0 - C : Carry Flag:

The carry flag C indicates a carry in an arithmetic or logic operation. See the Instruction Set Description for detailed information.

THE STACK POINTER - SP

An 8-bit register at I/O address $3D ($5D) forms the stack pointer of the AT90S2313. 8 bits are used to address the 128 bytes of SRAM in locations $60 - $DF.

Bit	7	6	5	4	3	2	1	0	
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when data is pushed onto the Stack with subroutine CALL and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt IRET.

Reset and Interrupt Handling

The AT90S2313 provides 10 different interrupt sources. These interrupts and the separate reset vector, each have a separate program vector in the program memory space. All the interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 2. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 - the External Interrupt Request 0, etc.

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT1	Timer/Counter1 Capture Event
5	$004	TIMER1 COMP1	Timer/Counter1 Compare Match
6	$005	TIMER1 OVF1	Timer/Counter1 Overflow
7	$006	TIMER0, OVF0	Timer/Counter0 Overflow
8	$007	UART, RX	UART, Rx Complete
9	$008	UART, UDRE	UART Data Register Empty
10	$009	UART, TX	UART, Tx Complete
11	$00A	ANA_COMP	Analog Comparator

The most typical and general program setup for the Reset and Interrupt Vector Addresses are:

Address	Labels	Code	Comments
$000		rjmp RESET	; Reset Handle
$001		rjmp EXT_INT0	; IRQ0 Handle
$002		rjmp EXT_INT1	; IRQ1 Handle
$003		rjmp TIM_CAPT1	; Timer1 capture Handle
$004		rjmp TIM_COMP1	; Timer1 compare Handle
$005		rjmp TIM_OVF1	; Timer1 overflow Handle
$006		rjmp TIM_OVF0	; Timer0 overflow Handle
$007		rjmp UART_RXC	; UART RX Complete Handle
$008		rjmp UART_DRE	; UDR Empty Handle
$009		rjmp UART_TXC	; UART TX Complete Handle
$00a		rjmp ANA_COMP	; Analog Comparator Handle
;			
$00b	MAIN:	<instr> xxx	; Main program start
…	…	…	

RESET SOURCES

The AT90S2313 has three sources of reset:

- Power-On Reset. The MCU is reset when a supply voltage is applied to the V_{CC} and GND pins.
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for more than two XTAL cycles
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are then set to their initial values, and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP - relative jump - instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 23 shows the reset logic. Table 3 defines the timing and electrical parameters of the reset circuitry.

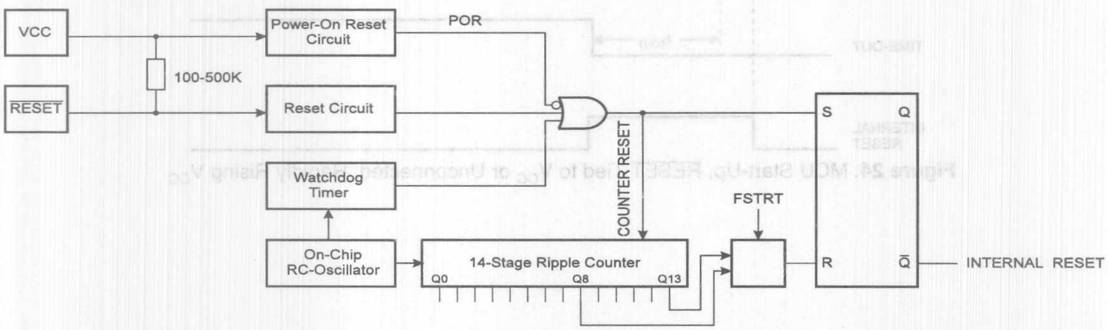


Figure 23. Reset Logic

Table 3. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-On Reset Threshold Voltage	1.8	2	2.2	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		$V_{CC}/2$		V
t_{POR}	Power-On Reset Period	2	3	4	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Unprogrammed	11	16	21	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Programmed	1.0	1.1	1.2	ms

POWER-ON RESET

A Power-On Reset (POR) circuit ensures that the device is not started until V_{CC} has reached a safe level. As shown in Figure 23, an internal timer is clocked from the Watchdog timer. This timer prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-On Threshold voltage - V_{POT}. See Figure 24 and Figure 25. The total reset period is the Power-On Reset period - t_{POR} + the Delay Time-out period - t_{TOUT}. The FSTRT fuse bit in the Flash can be programmed to give a shorter start-up time if a ceramic resonator or any other fast-start oscillator is used to clock the MCU.

As the $\overline{\text{RESET}}$ pin is pulled high by an on-chip resistor, the pin can be left unconnected if no external reset is required. Connecting $\overline{\text{RESET}}$ to V_{CC} will have the same effect. By holding the $\overline{\text{RESET}}$ pin low for a period after V_{CC} has been applied, the Power-On Reset period can be extended. Refer to Figure 26 for a timing example on this.

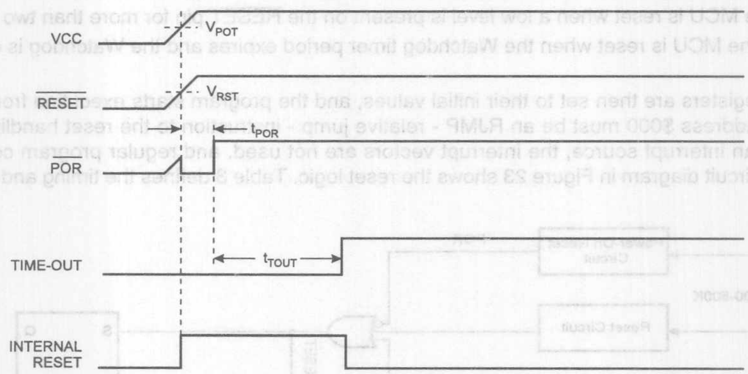


Figure 24. MCU Start-Up, $\overline{\text{RESET}}$ Tied to V_{CC} or Unconnected. Rapidly Rising V_{CC}

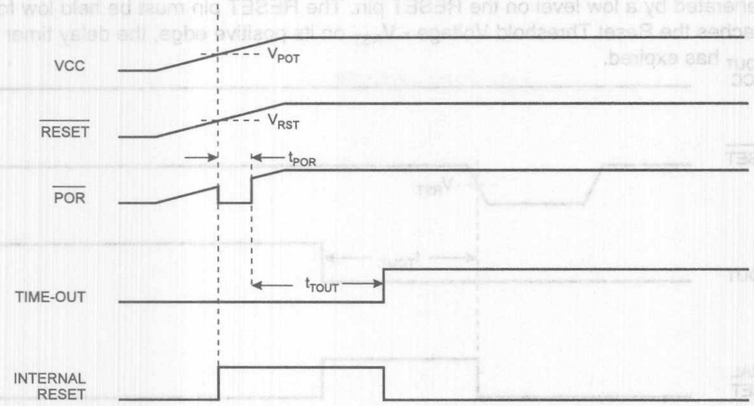


Figure 25. MCU Start-Up, RESET Tied to V_{CC} or Unconnected. Slowly Rising V_{CC}

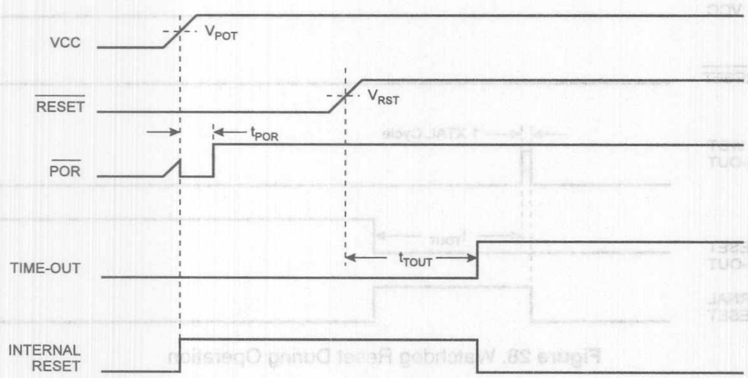


Figure 26. MCU Start-Up, RESET Controlled Externally

EXTERNAL RESET

An external reset is generated by a low level on the $\overline{RESET}$ pin. The $\overline{RESET}$ pin must be held low for at least two crystal clock cycles. When reaches the Reset Threshold Voltage - V_{RST} on its positive edge, the delay timer starts the MCU after the Time-out period t_{TOUT} has expired.

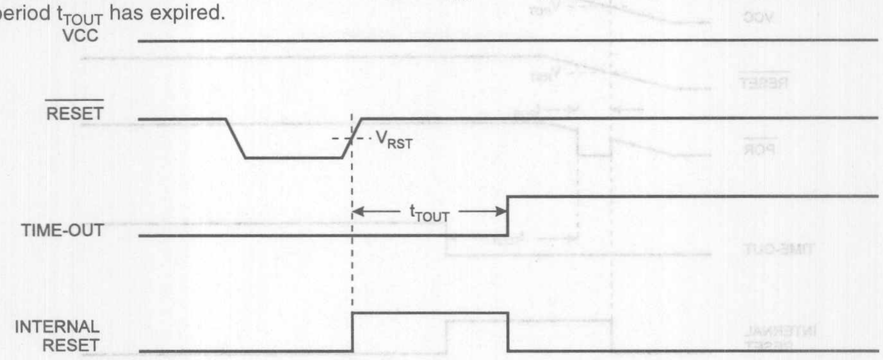


Figure 27. External Reset During Operation

WATCHDOG RESET

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to Page 3-39 for details on operation of the Watchdog.

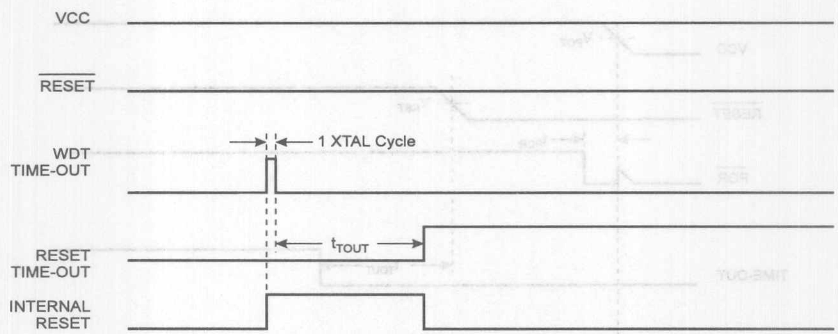


Figure 28. Watchdog Reset During Operation

INTERRUPT HANDLING

The AT90S2313 has two 8-bit Interrupt Mask control registers; GIMSK - General Interrupt Mask register and TIMSK - Timer/Counter Interrupt Mask register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software can set (one) the I-bit to enable interrupts. The I-bit is set (one) when a Return from Interrupt instruction - RETI - is executed.

For Interrupts triggered by events that can remain static (e.g. the Output Compare register1 matching the value of Timer/Counter1) the interrupt flag is set when the event occurs. If the interrupt flag is cleared and the interrupt condition persists, the flag will not be set until the event occurs the next time.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

THE GENERAL INTERRUPT MASK REGISTER - GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INT1	INT0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INT1 : External Interrupt Request 1 Enable:
When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from program memory address $002. See also "external Interrupts".

Bit 6 - INT0 : External Interrupt Request 0 Enable:
When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from program memory address $001. See also "External Interrupts."

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and always read as zero.

THE GENERAL INTERRUPT FLAG REGISTER - GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INTF1 : External Interrupt Flag1:
When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $002. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bit 6 - INTF0 : External Interrupt Flag0:
When an event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $001. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and always read as zero.

THE TIMER/COUNTER INTERRUPT MASK REGISTER - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	TOIE1	OCIE1A	-	-	TICIE1	-	TOIE0	-	TIMSK
Read/Write	R/W	R/W	R	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOIE1 : Timer/Counter1 Overflow Interrupt Enable:
When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt (at vector $005) is executed if an overflow in Timer/Counter1 occurs. The Overflow Flag (Timer/Counter1) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR. When Timer/Counter1 is in PWM mode, the Timer Overflow flag is set when the counter changes counting direction at $0000.

Bit 6 - OCIE1A :Timer/Counter1 Output Compare Match Interrupt Enable:
When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Compare Match interrupt is enabled. The corresponding interrupt (at vector $004) is executed if a Compare match in Timer/Counter1 occurs. The Compare Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 5,4 - Res :Reserved bits:
These bits are reserved bits in the AT90S2313 and always read zero.

Bit 3 - TICIE1 : Timer/Counter1 Input Capture Interrupt Enable:
When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Input Capture Event Interrupt is enabled. The corresponding interrupt (at vector $003) is executed if a capture-triggering event occurs on pin 11, PD6(ICP). The Input Capture Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S2313 and always reads as zero.

Bit 1 - TOIE0 : Timer/Counter0 Overflow Interrupt Enable:
When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt (at vector $006) is executed if an overflow in Timer/Counter0 occurs. The Overflow Flag (Timer0) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 0 - Res : Reserved bit:
This bit is a reserved bit in the AT90S2313 and always reads as zero.

THE TIMER/COUNTER INTERRUPT FLAG REGISTER - TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	TOV1	OCF1A	-	-	ICF1	-	TOV0	-	TIFR
Read/Write	R/W	R/W	R	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOV1 : Timer/Counter1 Overflow Flag:
The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logical one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

Bit 6 - OCF1A : Output Compare Flag 1A:
The OCF1A bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1A - Output Compare Register1 A. OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared by writing a logical one to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare match Interrupt Enable), and the OCF1A is set (one), the Timer/Counter1 Compare match Interrupt is executed.

Bits 5, 4 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and always read zero.

Bit 3 - ICF1 : - Input Capture Flag 1:
The ICF1 bit is set (one) to flag an input capture event, indicating that the Timer/Counter1 value has been transferred to the input capture register - ICR1. ICF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ICF1 is cleared by writing a logical one to the flag.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S2313 and always reads zero.

Bit 1 - TOV0 : Timer/Counter0 Overflow Flag:

The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logical one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

Bit 0 - Res : Reserved bit:

This bit is a reserved bit in the AT90S2313 and always reads zero.

EXTERNAL INTERRUPTS

The external interrupts are triggered by the INT1 and INT0 pins.Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register - MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

The external interrupts are set up as described in the specification for the MCU Control Register - MCUCR.

INTERRUPT RESPONSE TIME

The interrupt execution response for all the enabled AVR interrupts is 4 clock cycles minimum. After the 4 clock cycles the program vector address for the actual interrupt handling routine is executed. During this 4 clock cycle period, the Program Counter (2 bytes) is pushed onto the Stack, and the Stack Pointer is decremented by 2. The vector is a relative jump to the interrupt routine, and this jump takes 2 clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine takes 4 clock cycles. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, and the Stack Pointer is incremented by 2. When *AVR* exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register - SREG - is not handled by the *AVR* hardware, neither for interrupts nor for subroutines. For the routines requiring a storage of the SREG, this must be performed by user software.

THE MCU CONTROL REGISTER - MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	-	-	SE	SM	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R	R	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7, 6 - Res : Reserved bits:

These bits are reserved bits in the AT90S2313 and always read as zero.

Bit 5 - SE : Sleep Enable:

The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

Bit 4 - SM : Sleep Mode:

This bit selects between the two available sleep modes. When SM is cleared (zero), Idle Mode is selected as Sleep Mode. When SM is set (one), Power Down mode is selected as sleep mode. For details, refer to the paragraph "Sleep Modes" below.

Bits 3, 2 - ISC11, ISC10 : Interrupt Sense Control 1 bit 1 and bit 0:

The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask in the GIMSK register is set. The level and edges on the external INT1 pin that activate the interrupt are defined in the following table:

Table 4. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Note: When changing the ISC11/ISC10 bits, INT1 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bits 1, 0 - ISC01, ISC00 : Interrupt Sense Control 0 bit 1 and bit 0:

The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask is set. The level and edges on the external INT0 pin that activate the interrupt are defined in the following table:

Table 5.. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Notes: When changing the ISC10/ISC00 bits, INT0 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Sleep Modes

To enter the sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM and I/O memory are unaltered. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset vector.

Note that if a *level* triggered interrupt is used for wake-up from power down, the low level must be held for a time longer than the oscillator start-up time of 16 ms. Otherwise, the interrupt flag may return to zero before the MCU starts executing.

IDLE MODE

When the SM bit is cleared (zero), the SLEEP instruction forces the MCU into the Idle Mode stopping the CPU but allowing Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like Timer Overflow interrupt and watchdog reset. If wakeup from the Analog Comparator interrupt is not required, the analog comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status register - ACSR. This will reduce power consumption during Idle Mode.

POWER DOWN MODE

When the SM bit is set (one), the SLEEP instruction forces the MCU into the Power Down Mode. In this mode, the external oscillator is stopped. The user can select whether the watchdog shall be enabled during power-down mode. If the watchdog is enabled, it will wake up the MCU when the Watchdog Time-out period expires. If the watchdog is disabled, only an external reset or an external level triggered interrupt can wake up the MCU.

Timer / Counters

The AT90S2313 provides two general purpose Timer/Counters - one 8-bit T/C and one 16-bit T/C. The Timer/Counters have individual prescaling selection from the same 10-bit prescaling timer. Both Timer/Counters can either be used as a timer with an internal clock timebase or as a counter with an external pin connection which triggers the counting.

The Timer/Counter Prescaler

Figure 29 shows the general Timer/Counter prescaler.

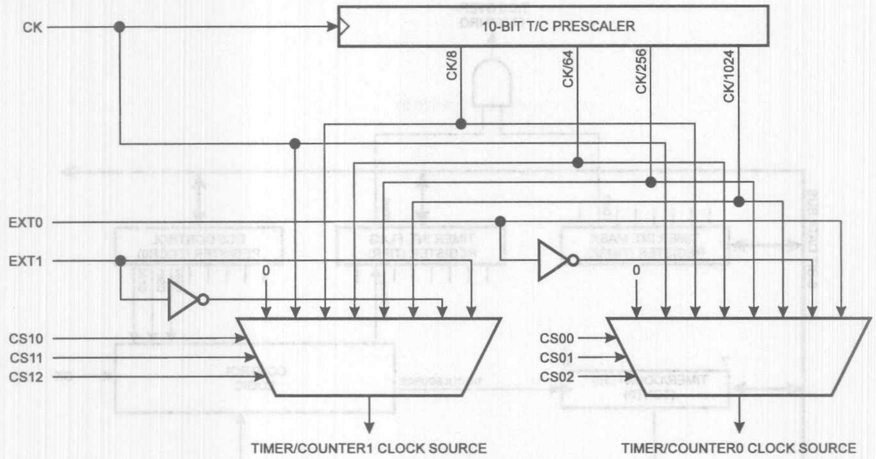


Figure 29. Timer/Counter Prescaler

The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. For the two Timer/Counters, added selections as CK, external source and stop, can be selected as clock sources.

The 8-Bit Timer/Counter0

Figure 30 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter0 Control Register - TCCR0. The overflow status flag is found in the Timer/Counter Interrupt Flag Register - TIFR. Control signals are found in the Timer/Counter0 Control Register - TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

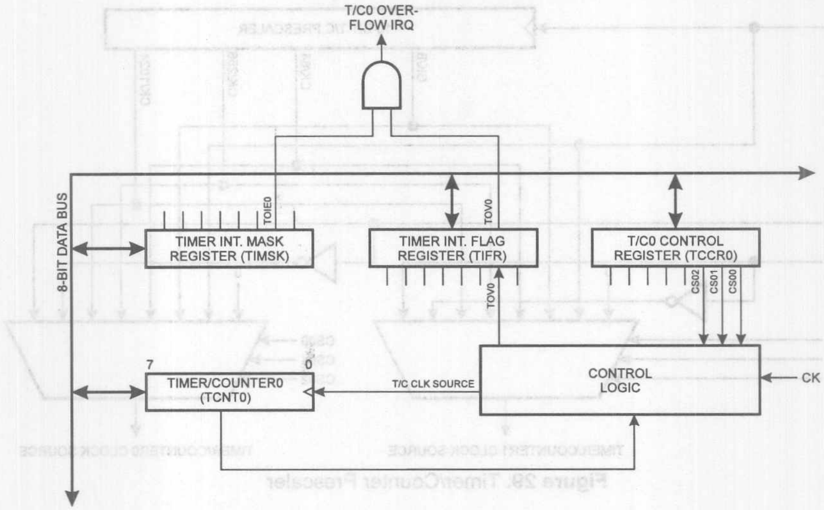


Figure 30. Timer/Counter 0 Block Diagram

THE TIMER/COUNTER0 CONTROL REGISTER - TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	-	-	-	-	-	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..3 - Res : Reserved bits:

These bits are reserved bits in the AT90S2313 and always read zero.

Bits 2,1,0 - CS02, CS01, CS00 : Clock Select0, bit 2,1 and 0:

The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer0.

Table 6. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual data direction control register (cleared to zero gives an input pin).

THE TIMER COUNTER 0 - TCNT0

Bit	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the timer clock cycle following the write operation.

The 16-Bit Timer/Counter1

Figure 31 shows the block diagram for Timer/Counter1.

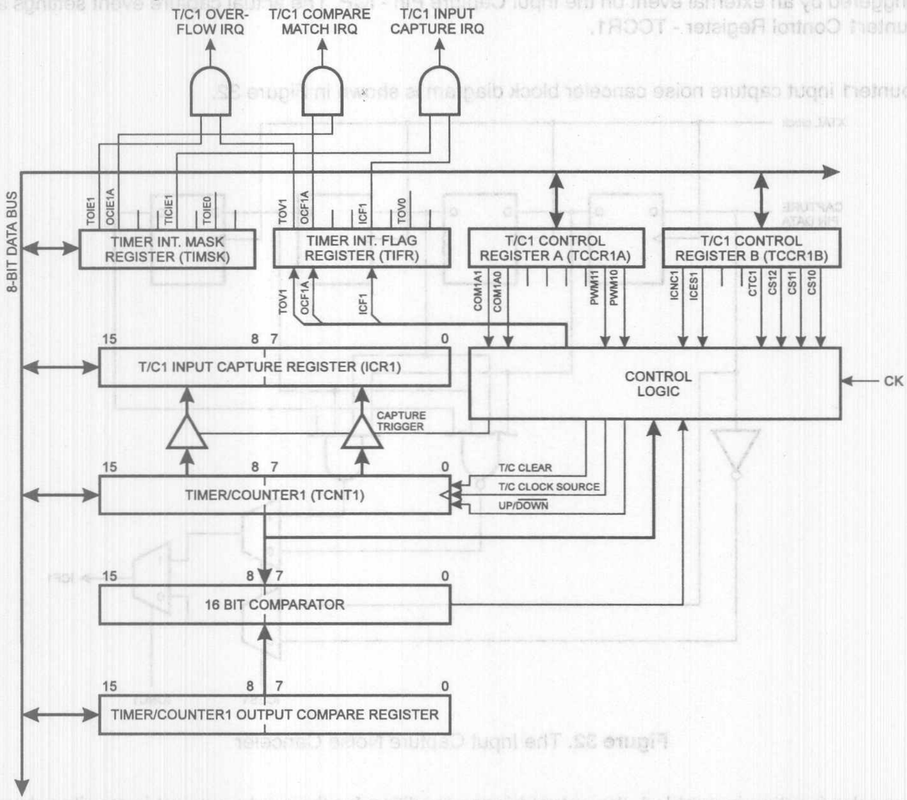


Figure 31. Timer/Counter1 Block Diagram

The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter1 Control Register - TCCR1A. The different status flags (overflow, compare match and capture event) and control signals are found in the Timer/Counter Interrupt Flag Register - TIFR. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 16-bit Timer/Counter1 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports an Output Compare function using the Output Compare Register 1A - OCR1A as the data source to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compare matches, and actions on the Output Compare pin 1 on compare matches.

Timer/Counter1 can also be used as a 8, 9 or 10-bit Pulse With Modulator. In this mode the counter and the OCR1 register serve as a glitch-free stand-alone PWM with centered pulses. Refer to Page 3-38 for a detailed description on this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register - ICR1, triggered by an external event on the Input Capture Pin - ICP. The actual capture event settings are defined by the Timer/Counter1 Control Register - TCCR1.

The Timer/Counter1 input capture noise canceler block diagram is shown in Figure 32.

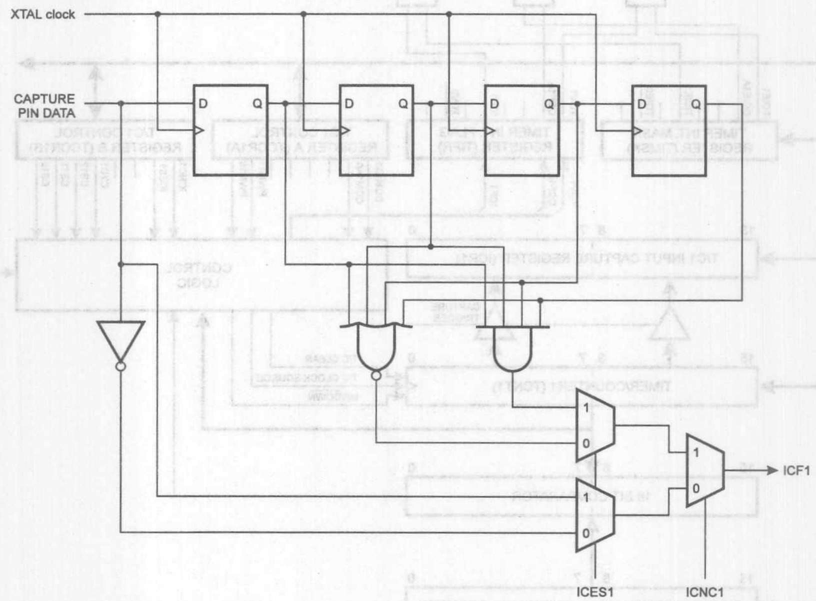


Figure 32. The Input Capture Noise Canceler

If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over 4 samples before the capture is activated. The input pin signal is sampled at XTAL clock frequency.

THE TIMER/COUNTER1 CONTROL REGISTER A - TCCR1A

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	-	-	-	-	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R	R	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7,6 - COM1A1, COM1A0 : Compare Output Mode1, bits 1 and 0:
The COM1A1 and COM1A0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1 - Output Compare pin 1. Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 7.

Table 7. Compare 1 Mode Select

COM1A1	COM1A0	Description
0	0	Timer/Counter1 disconnected from output pin OC1
0	1	Toggle the OC1 output line.
1	0	Clear the OC1 output line (to zero).
1	1	Set the OC1 output line (to one).

In PWM mode, these bits have a different function. Refer to Table 11 for a detailed description.

When changing the COM1A1/COM1A0 bits, Output Compare Interrupt 1A must be disabled by clearing its Interrupt Enable bit in the TIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bits 5..2 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and always read zero.

Bits 1,0 - PWM11, PWM10 : Pulse Width Modulator Select Bits:
These bits select PWM operation of Timer/Counter1 as specified in Table 8 . This mode is described on Page 3-38.

Table 8. PWM Mode Select

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

THE TIMER/COUNTER1 CONTROL REGISTER B - TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ICNC1 : Input Capture1 Noise Canceler (4 CKs):
When the ICNC1 bit is cleared (zero), the input capture trigger noise canceler function is disabled. The input capture is triggered at the first rising/falling edge sampled on the ICP - input capture pin - as specified. When the ICNC1 bit is set (one), four successive samples are measures on the ICP - input capture pin, and all samples must be high/low according to the input capture trigger specification in the ICES1 bit. The actual sampling frequency is the XTAL clock frequency.

Bit 6 - ICES1 : Input Capture1 Edge Select:
While the ICES1 bit is cleared (zero), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the falling edge of the input capture pin - ICP. While the ICES1 bit is set (one), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the rising edge of the input capture pin - ICP.

Bits 5, 4 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and always read zero.

Bit 3 - CTC1 : Clear Timer/Counter1 on Compare match:
When the CTC1 control bit is set (one), the Timer/Counter1 is reset to $0000 in the clock cycle after a compare match. If the CTC1 control bit is cleared, the Timer/Counter1 continues counting until it is stopped, cleared, wraps around (overflow) or changes direction. In PWM mode, this bit has no effect.

Bits 2,1,0 - CS12, CS11, CS10 : Clock Select1, bit 2,1 and 0:
The Clock Select1 bits 2,1 and 0 define the prescaling source of Timer/Counter1.

Table 9. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual direction control register (cleared to zero gives an input pin).

THE TIMER/COUNTER1 - TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP).

- **TCNT1 Timer/Counter1 Write:**

  When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP register, and all 16 bits are written to the TCNT1 Timer/Counter1 register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.

- **TCNT1 Timer/Counter1 Read:**

  When the CPU reads the low byte TCNT1L, the data of the low byte TCNT1L is sent to the CPU and the data of the high byte TCNT1H is placed in the TEMP register. When the CPU reads the data in the high byte TCNT1H, the CPU receives the data in the TEMP register. Consequently, the low byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

TIMER/COUNTER1 OUTPUT COMPARE REGISTER A - OCR1AH AND OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The output compare register is a 16-bit read/write register.

The Timer/Counter1 Output Compare Register contains the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status register.

Since the Output Compare Register - OCR1A - is a 16-bit register, a temporary register TEMP is used when OCR1A is written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH, the data is temporarily stored in the TEMP register. When the CPU writes the low byte, OCR1AL, the TEMP register is simultaneously written to OCR1AH. Consequently, the high byte OCR1AH must be written first for a full 16-bit register write operation.

THE TIMER/COUNTER1 INPUT CAPTURE REGISTER - ICR1H AND ICR1L

Bit	15	14	13	12	11	10	9	8	
$25 ($45)	MSB								ICR1H
$24 ($44)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The input capture register is a 16-bit read-only register.

When the rising or falling edge (according to the input capture edge setting - ICES1) of the signal at the input capture pin - ICP - is detected, the current value of the Timer/Counter1 is transferred to the Input Capture Register - ICR1. At the same time, the input capture flag - ICF1 - is set (one).

Since the Input Capture Register - ICR1 - is a 16-bit register, a temporary register TEMP is used when ICR1 is read to ensure that both bytes are read simultaneously. When the CPU reads the low byte ICR1L, the data is sent to the CPU and the data of the high byte ICR1H is placed in the TEMP register. When the CPU reads the data in the high byte ICR1H, the CPU receives the data in the TEMP register. Consequently, the low byte ICR1L must be accessed first for a full 16-bit register read operation.

TIMER/COUNTER1 IN PWM MODE

When the PWM mode is selected, Timer/Counter1 and the Output Compare Register1 - OCR1A, form a 8, 9 or 10-bit, free-running, glitch-free and phase correct PWM with output on the PB3(OC1) pin. Timer/Counter1 acts as an up/down counter, counting up from $0000 to TOP (see Table 10), when it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the 8, 9 or 10 least significant bits of OCR1A, the PD1(OC1) pin is set or cleared according to the settings of the COM1A1 and COM1A0 bits in the Timer/Counter1 Control Register TCCR1. Refer to Table 11 for details.

Table 10. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP value	Frequency
8-bit	$00FF (255)	$f_{TC1}/510$
9-bit	$01FF (511)	$f_{TC1}/1022$
10-bit	$03FF(1023)	$f_{TC1}/2046$

Table 11. Compare1 Mode Select in PWM Mode

COM1A1	COM1A0	Effect on OC1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, upcounting. Set on compare match, downcounting (non-inverted PWM).
1	1	Cleared on compare match, downcounting. Set on compare match, upcounting (inverted PWM).

Note that in the PWM mode, the 10 least significant OCR1A bits, when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A write. See Figure 33 for an example.

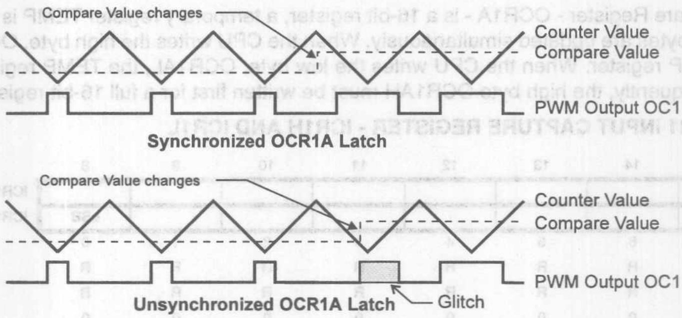


Figure 33. Effects on Unsynchronized OCR1 Latching

When OCR1A contains $0000 or TOP, the output OC1 is held low or high according to the settings of COM1A1 and COM1A0. This is shown in Table 12.

Table 12. PWM Outputs OCR = $0000 or TOP

COM1A1	COM1A0	OCR1A	Output OC1
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

In PWM mode, the Timer Overflow Flag1, TOV1, is set when the counter changes direction at $0000. Timer Overflow Interrupt1 operates exactly as in normal Timer/Counter mode, i.e. it is executed when TOV1 is set provided that Timer Overflow Interrupt1 and global interrupts are enabled. This does also apply to the Timer Output Compare1 flag and interrupt.

The Watchdog Timer

The Watchdog Timer is clocked from a separate on-chip oscillator which runs at 1MHz This is the typical value at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted from 16 to 2048 ms. The WDR - Watchdog Reset - instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog reset, the AT90S2313 resets and executes from the reset vector. For timing details on the Watchdog reset, refer to Page 3-26.

To prevent unintentional disabling of the watchdog, a special turn-off secuence must be followed when the watchdog is disabled.Refer to the description of the Watchdog Timer Control Regiter for details.

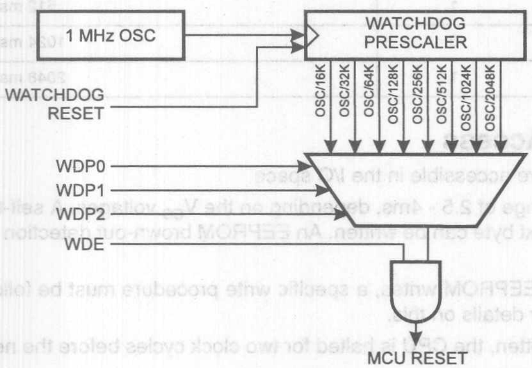


Figure 34. Watchdog Timer

THE WATCHDOG TIMER CONTROL REGISTER - WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	-	-	-	WDTTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..5 - Res : Reserved bits:

These bits are reserved bits in the AT90S2313 and will always read as zero.

Bit 4 - WDTOE : Watch Dog Turn-Off Enable:
This bit must be set (one) when the WDE bit is cleared. Otherwise, the watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure.

Bit 3 - WDE : Watch Dog Enable:
When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logcal one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logical 0 to WDE. This disables the watchdog.

Bits 2..0 - WDP2, WDP1, WDP0 : Watchdog Timer Prescaler 1 and 0:
The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding timeout periods are shown in Table 13.

Table 13. Watch Dog Timer Prescale Select (Typical Values at V_{CC} = 5.0V)

WDP2	WDP1	WDP0	Timeout Period
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written. An EEPROM brown-out detection prevents writing to the EEPROM if V_{CC} is below a certain level.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read or written, the CPU is halted for two clock cycles before the next instruction is executed.

THE EEPROM ADDRESS REGISTER - EEAR

Bit	7	6	5	4	3	2	1	0	
$1E ($3E)	-	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - Res : Reserved bit:
This bit is a reserved bit in the AT90S2313 and will always read as zero.

Bit 6..0 - EEAR6..0 : EEPROM Address:
The EEPROM Address Register - EEAR6..0 - specifies the EEPROM address in the 128 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 127.

THE EEPROM DATA REGISTER - EEDR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7..0 - EEDR7..0 : EEPROM Data:
For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

THE EEPROM CONTROL REGISTER - EECR

Bit	7	6	5	4	3	2	1	0	
$1C ($3C)	-	-	-	-	-	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7..3 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and will always read as zero.

Bit 2 - EEMWE : EEPROM Master Write Enable:
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for a EEPROM write procedure.

Bit 1 - EEWE : EEPROM Write Enable:
The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is unessential):

1. Wait until EEWE becomes zero.
2. Write new EEPROM address to EEAR (optional)
3. Write new EEPROM data to EEDR (optional)
4. Write a logical one to the EEMWE bit in EECR
5. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

When the write access time (typically 2.5 ms at V_{CC} = 5V or 4 ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 - EERE : EEPROM Read Enable:
The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress when new data or address is written to the EEPROM I/O registers, the write operation will be interrupted, and the result is undefined.

The UART

The AT90S2313 features a full duplex Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- Baud rate generator generates any baud rate
- High baud rates at low XTAL frequencies
- 8 or 9 bits data
- Noise filtering
- Overrun detection
- Framing Error detection
- False Start Bit detection
- Three separate interrupts on TX Complete, TX Data Register Empty and RX Complete

Data Transmission

A block schematic of the UART transmitter is shown in Figure 35.

Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDR. Data is transferred from UDR to the Transmit shift register when:

- A new character has been written to UDR after the stop bit from the previous character has been shifted out. The shift register is loaded immediately.
- A new character has been written to UDR before the stop bit from the previous character has been shifted out. The shift register is loaded when the stop bit of the character currently being transmitted has been shifted out.

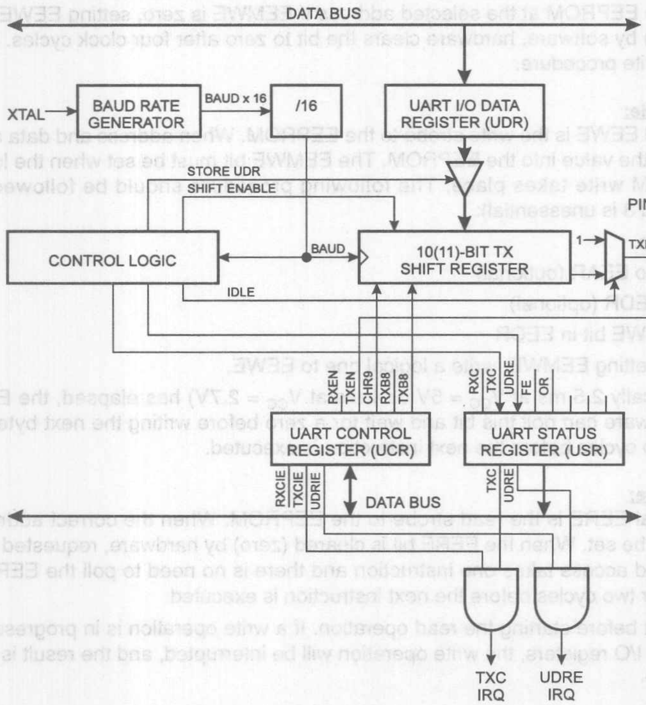


Figure 35. UART Transmitter

At this time the UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the data is transferred from UDR to the 10(11)-bit shift register, bit 0 of the shift register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit shift register.

On the Baud Rate clock following the transfer operation to the shift register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the shift register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR register to send when the stop bit is shifted out, the UDRE flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the TX Complete Flag, TXC, in USR is set.

The TXEN bit in UCR enables the UART transmitter when set (one). By clearing this bit (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to the PD1 pin regardless of the setting of the DDD1 bit in DDRD.

Data Reception

Figure 36 shows a block diagram of the UART Receiver

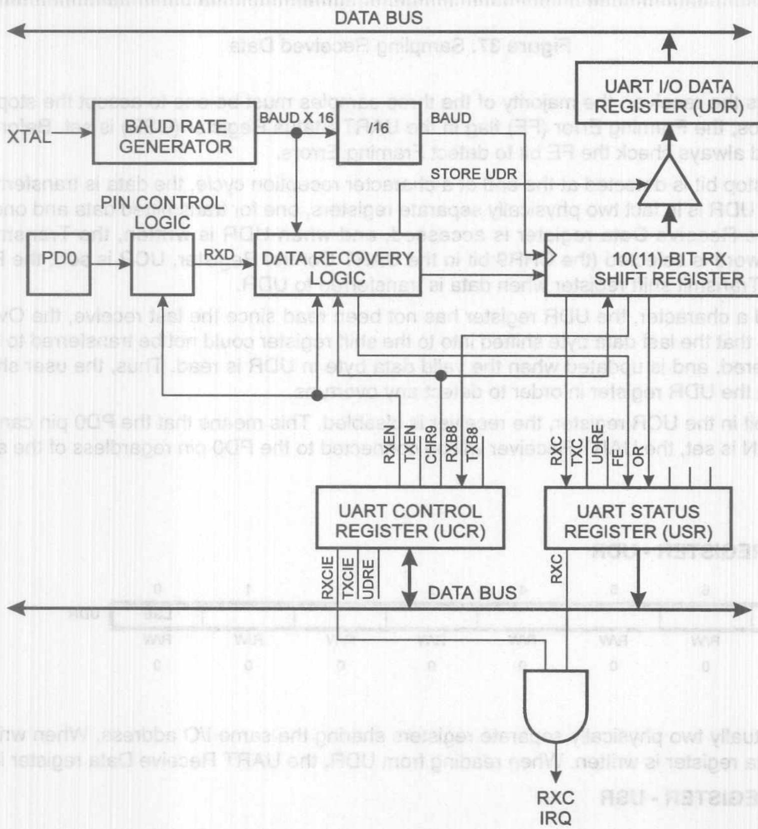


Figure 36. UART Receiver

The receiver front-end logic samples the signal on the RXD pin at a frequency 16 times the baud rate. While the line is idle, one single sample of logical zero will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1 to 0-transition, the receiver samples the RXD pin at samples 8, 9 and 10. If two or more of these three samples are found to be logical ones, the start bit is rejected as a noise spike and the receiver starts looking for the next 1 to 0-transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 8, 9 and 10. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the transmitter shift register as they are sampled. Sampling of an incoming character is shown in Figure 37.

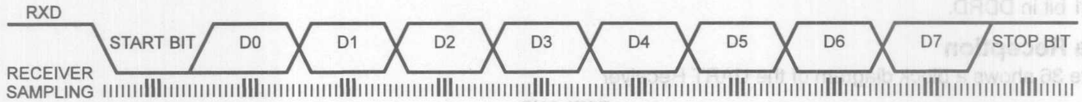


Figure 37. Sampling Received Data

When the stop bit enters the receiver, the majority of the three samples must be one to accept the stop bit. If two or more samples are logical zeros, the Framing Error (FE) flag in the UART Status Register (USR) is set. Before reading the UDR register, the user should always check the FE bit to detect Framing Errors.

Whether or not a valid stop bit is detected at the end of a character reception cycle, the data is transferred to UDR and the RXC flag in USR is set. UDR is in fact two physically separate registers, one for transmitted data and one for received data. When UDR is read, the Receive Data register is accessed, and when UDR is written, the Transmit Data register is accessed. If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the RXB8 bit in UCR is loaded with bit 9 in the Transmit shift register when data is transferred to UDR.

If, after having received a character, the UDR register has not been read since the last receive, the OverRun (OR) flag in UCR is set. This means that the last data byte shifted into to the shift register could not be transferred to UDR and has been lost. The OR bit is buffered, and is updated when the valid data byte in UDR is read. Thus, the user should always check the OR bit after reading the UDR register in order to detect any overruns.

By clearing the RXEN bit in the UCR register, the receiver is disabled. This means that the PD0 pin can be used as a general I/O pin. When RXEN is set, the UART Receiver will be connected to the PD0 pin regardless of the setting of the DDD0 bit in DDRD.

UART Control

THE UART I/O DATA REGISTER - UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UDR register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDR, the UART Receive Data register is read.

THE UART STATUS REGISTER - USR

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	OR	-	-	-	USR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	1	0	0	0	0	0	

The USR register is a read-only register providing information on the UART Status.

Bit 7 - RXC: UART Receive Complete:
This bit is set (one) when a received character is transferred from the Receiver Shift register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set(one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

Bit 6 - TXC : UART Transmit Complete:
This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical one to the bit.

Bit 5 - UDRE : UART Data Register Empty:
This bit is set (one) when a character written to UDR is transferred to the Transmit shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCR is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.

Bit 4 - FE : Framing Error:
This bit is set if a Framing Error condition is detected, i.e. when the stop bit of an incoming character is zero.

The FE bit is cleared when the stop bit of received data is one.

Bit 3 - OR : OverRun:
This bit is set if an Overrun condition is detected, i.e. when a character already present in the UDR register is not read before the next character has been shifted into the Receiver Shift register. The OR bit is buffered, which means that it will be set once the valid data still in UDRE is read.

The OR bit is cleared (zero) when data is received and transferred to UDR.

Bits 2..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and will always read as zero.

THE UART CONTROL REGISTER - UCR

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - RXCIE : RX Complete Interrupt Enable:
When this bit is set (one), a setting of the RXC bit in USR will cause the Receive Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 6 - TXCIE : TX Complete Interrupt Enable:
When this bit is set (one), a setting of the TXC bit in USR will cause the Transmit Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 5 - UDRIE : UART Data Register Empty Interrupt Enable:
When this bit is set (one), a setting of the UDRE bit in USR will cause the UART Data Register Empty interrupt routine to be executed provided that global interrupts are enabled.

Bit 4 - RXEN : Receiver Enable:
This bit enables the UART receiver when set (one). When the receiver is disabled, the TXC, OR and FE status flags cannot become set. If these flags are set, turning off RXEN does not cause them to be cleared.

Bit 3 - TXEN : Transmitter Enable:
This bit enables the UART transmitter when set (one). When disabling the transmitter while transmitting a character, the transmitter is not disabled before the character in the shift register plus any following character in UDR has been completely transmitted.

Bit 2 - CHR9 : 9 Bit Characters:
When this bit is set (one) transmitted and received characters are 9 bit long plus start and stop bits. The 9th bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The 9th data bit can be used as an extra stop bit or a parity bit.

Bit 1 - RXB8 : Receive Data Bit 8
When CHR9 is set (one), RXB8 is the 9th data bit of the received character.

Bit 0 - TXB8 : Transmit Data Bit 8
When CHR9 is set (one), TXB8 is the 9th data bit in the character to be transmitted.

THE BAUD RATE GENERATOR

The baud rate generator is a frequency divider which generates baud-rates according to the following equation:

$$\mathrm{BAUD} = \frac{f_{\mathrm{CK}}}{16(\mathrm{UBRR}+1)}$$

- BAUD = Baud-Rate
- f_{CK} = Crystal Clock frequency
- UBRR = Contents of the UART Baud Rate register, UBRR (0-255)

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBRR settings in Table 14. UBRR values which yield an actual baud rate differing less than 2% from the target baud rate, are bolded in the table.

Table 14. UBRR Settings at Various Crystal Frequencies

Baud Rate	1 MHz		%Error	1.8432 MHz		%Error	2 MHz		%Error	2.4576 MHz		%Error
2400	UBRR=	25	0.2	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	63	0.0
4800	UBRR=	12	0.2	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	31	0.0
9600	UBRR=	6	7.5	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	15	0.0
14400	UBRR=	3	7.8	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	10	3.1
19200	UBRR=	2	7.8	UBRR=	5	0.0	UBRR=	6	7.5	UBRR=	7	0.0
28800	UBRR=	1	7.8	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	6.3
57600	UBRR=	0	7.8	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	12.5
115200	UBRR=	0	84.3	UBRR=	0	0.0	UBRR=	0	7.8	UBRR=	0	25.0

Baud Rate	3.2768 MHz		%Error	3.6864 MHz		%Error	4 MHz		%Error	4.608 MHz		%Error
2400	UBRR=	84	0.4	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0
4800	UBRR=	42	0.8	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0
9600	UBRR=	20	1.6	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0
14400	UBRR=	13	1.6	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0
19200	UBRR=	10	3.1	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	14	0.0
28800	UBRR=	6	1.6	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0
57600	UBRR=	3	12.5	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0
115200	UBRR=	1	12.5	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	20.0

Baud Rate	7.3728 MHz		%Error	8 MHz		%Error	9.216 MHz		%Error	11.059 MHz		%Error
2400	UBRR=	191	0.0	UBRR=	207	0.2	UBRR=	239	0.0	UBRR=	287	-
4800	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0	UBRR=	143	0.0
9600	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0	UBRR=	71	0.0
14400	UBRR=	31	0.0	UBRR=	34	0.8	UBRR=	39	0.0	UBRR=	47	0.0
19200	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0	UBRR=	35	0.0
28800	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0	UBRR=	23	0.0
57600	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0	UBRR=	11	0.0
115200	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0	UBRR=	5	0.0

Baud Rate	14.746 MHz		%Error	16 MHz		%Error	18.432 MHz		%Error	20 MHz		%Error
2400	UBRR=	383	-	UBRR=	416	-	UBRR=	479	-	UBRR=	520	-
4800	UBRR=	191	0.0	UBRR=	207	0.2	UBRR=	239	0.0	UBRR=	259	-
9600	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0	UBRR=	129	0.2
14400	UBRR=	63	0.0	UBRR=	68	0.6	UBRR=	79	0.0	UBRR=	86	0.2
19200	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0	UBRR=	64	0.2
28800	UBRR=	31	0.0	UBRR=	34	0.8	UBRR=	39	0.0	UBRR=	42	0.9
57600	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0	UBRR=	21	1.4
115200	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0	UBRR=	10	1.4

THE UART BAUD RATE REGISTER - UBRR

Bit	7	6	5	4	3	2	1	0	
\$09 (\$29)	MSB							LSB	UBRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UBRR register is an 8-bit read/write register which specifies the UART Baud Rate according to the formula on the previous page.

The Analog Comparator

The analog comparator compares the input values on the positive pin AIN0 (PB0) and the negative pin PB1(AIN1). When the voltage on the positive pin PB0 (AIN0) is higher than the voltage on the negative PB1 (AIN1), the Analog Comparator Output, ACO is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 38.

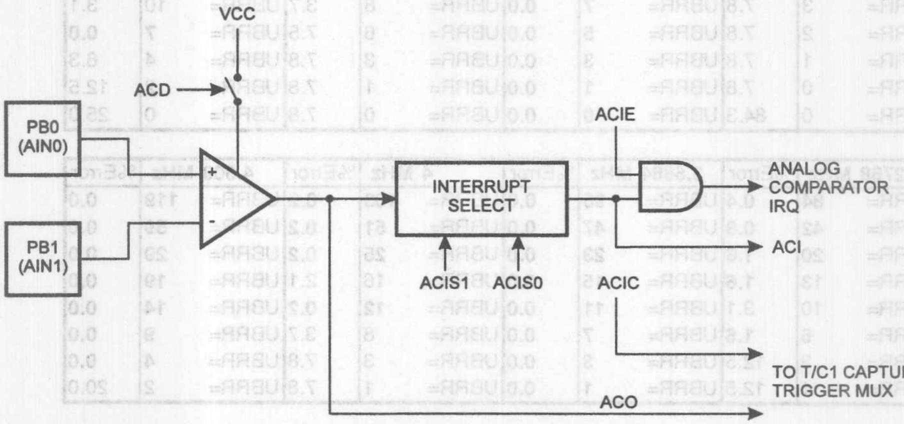


Figure 38. Analog Comparator Block Diagram

THE ANALOG COMPARATOR CONTROL AND STATUS REGISTER - ACSR

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ACD : Analog Comparator Disable

When this bit is set (one), the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. It is most commonly used if power consumption during Idle Mode is critical, and wake-up from the analog comparator is not required. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 - Res : Reserved bit:

This bit is a reserved bit in the AT90S2313 and will always read as zero.

Bit 5 - ACO : Analog Comparator Output:

ACO is directly connected to the comparator output.

Bit 4 - ACI : Analog Comparator Interrupt Flag:

This bit is set (one) when a comparator output event triggers the interrupt mode defined by ACIS1 and ACIS0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set (one) and the I-bit in SREG is set (one). ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 - ACIE : Analog Comparator Interrupt Enable:
When the ACIE bit is set (one) and the I-bit in the Status Register is set (one), the analog comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

Bit 2 - ACIC : Analog Comparator Input Capture enable:
When set (one), this bit enables the Input Capture function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When cleared (zero), no connection between the analog comparator and the Input Capture function is given. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set (one).

Bits 1.0 - ACIS1, ACIS0 : Analog Comparator Interrupt Mode Select:
These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 15.

Table 15. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

Note: When changing the ACIS1/ACIS0 bits, The Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

I/O-Ports

Port B

Port B is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port B, one each for the Data Register - PORTB, $18 ($38), Data Direction Register - DDRB, $17($37) and the Port B Input Pins - PINB, $16($36). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The Port B output buffers can sink 20mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The Port B pins with alternate functions are shown in the following table:

Table 16. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	AIN0 (Analog comparator positive input)
PB1	AIN1 (Analog comparator negative input)
PB3	OC1 (Timer/Counter1 Output compare match output)
PB5	MOSI (Data input line for memory downloading)
PB6	MISO (Data output line for memory uploading)
PB7	SCK (Serial clock input)

When the pins are used for the alternate function the DDRB and PORTB register has to be set according to the alternate function description.

THE PORT B DATA REGISTER - PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B DATA DIRECTION REGISTER - DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B INPUT PINS ADDRESS - PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port B Input Pins address - PINB - is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the PORTB Data Latch is read, and when reading PINB, the logical values present on the pins are read.

PORTB AS GENERAL DIGITAL I/O

All 8 bits in port B are equal when used as digital I/O pins.

PBn, General I/O pin: The DDBn bit in the DDRB register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 17. DDBn Effects on Port B Pins

DDBn	PORTBn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PBn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

ALTERNATE FUNCTIONS OF PORTB

The alternate pin functions of Port B are:

<u>SCK - PORTB, Bit 7:</u>
SCK , Clock input pin for Memory up/downloading.

<u>MISO - PORTB, Bit 6:</u>
MISO, Data output pin for Memory uploading.

<u>MOSI - PORTB, Bit 5:</u>
MOSI , Data input pin for Memory downloading.

OC1 - PORTB, Bit 3:
OC1, Output compare match output: The PB3 pin can serve as an external output when timer 1 compare match. The PB3 pin has to be configured as an output (DDB3 is set (one)) to serve this function. See the timer description for further details, and how to enable the output.

AIN1 - PORTB, Bit 1:
AIN1, Analog Comparator Negative Input. When configured as an input (DDB1 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB1 is cleared (zero)), this pin also serves as the negative input of the on-chip analog comparator.

AIN0 - PORTB, Bit 0:
AIN0, Analog Comparator Positive Input. When configured as an input (DDB0 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB0 is cleared (zero)), this pin also serves as the positive input of the on-chip analog comparator.

3

PORTB SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

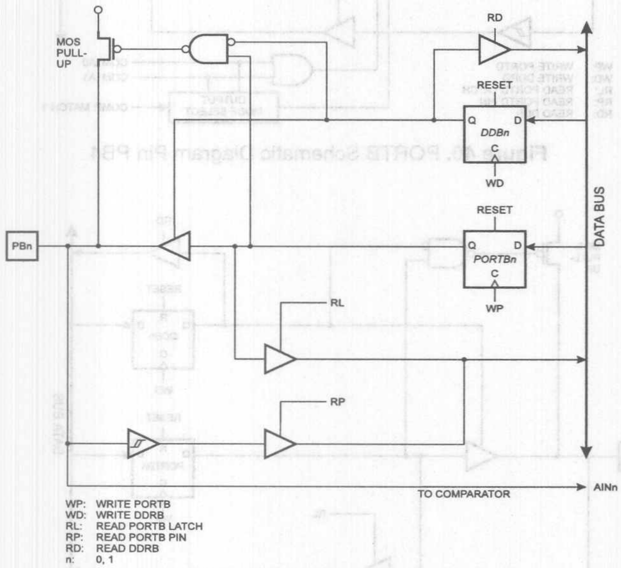


Figure 39. PORTB Schematic Diagram (pins PB0 and PB1)

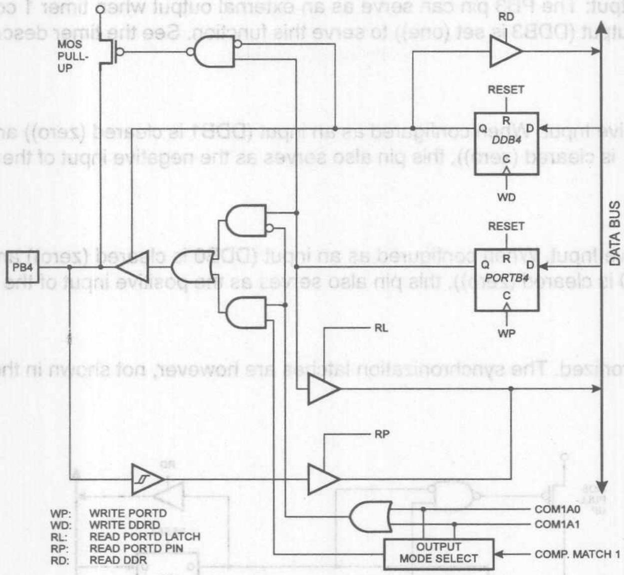


Figure 40. PORTB Schematic Diagram Pin PB4

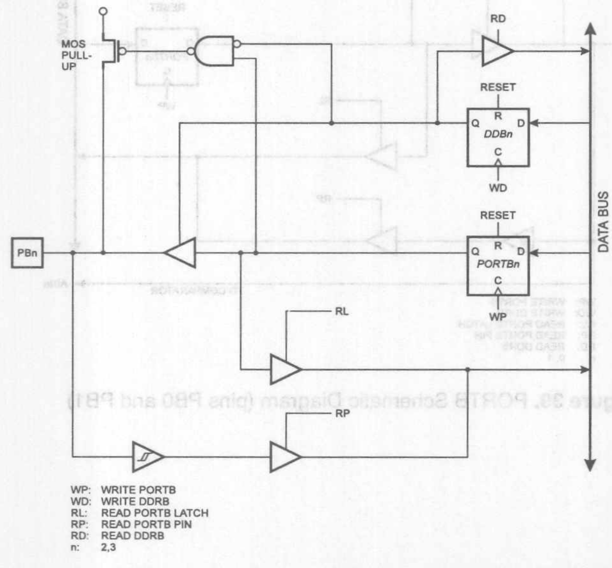


Figure 41. PORTB Schematic Diagram, Pins PB2 and PB3

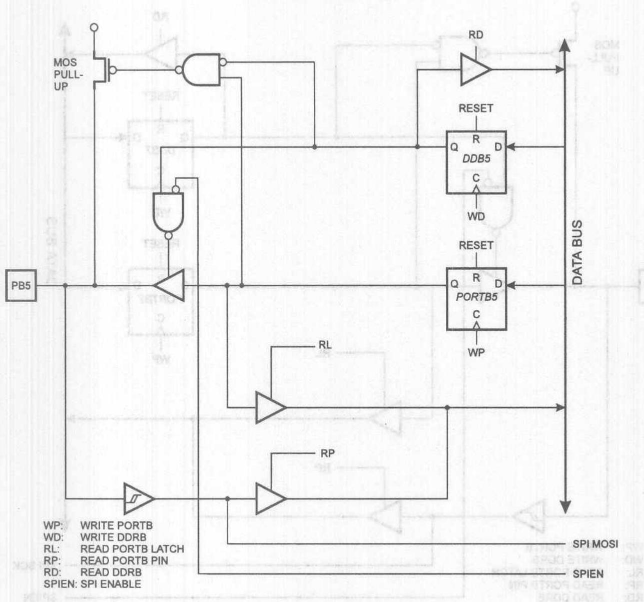


Figure 42. PORTB Schematic Diagram Pin PB5

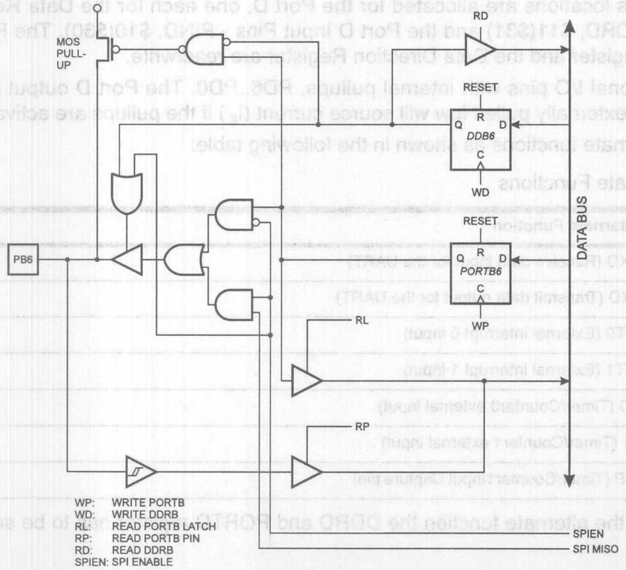


Figure 43. PORTB Schematic Diagram, Pin PB6

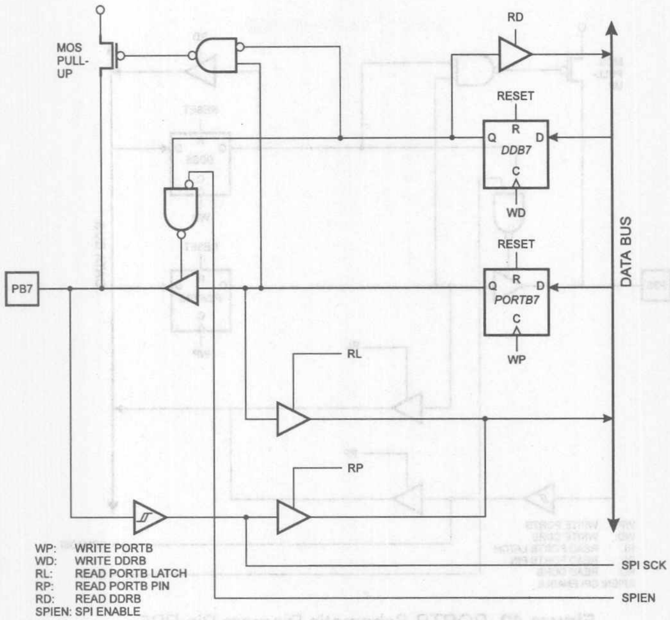


Figure 44. PORTB Schematic Diagram, Pin PB7

Port D

Three data memory address locations are allocated for the Port D, one each for the Data Register - PORTD, $12($32), Data Direction Register - DDRD, $11($31) and the Port D Input Pins - PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

Port D has seven bi-directional I/O pins with internal pullups, PD6..PD0. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Some Port D pins have alternate functions as shown in the following table:

Table 18. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RXD (Receive data input for the UART)
PD1	TXD (Transmit data output for the UART)
PD2	INT0 (External interrupt 0 input)
PD3	INT1 (External interrupt 1 input)
PD4	TO (Timer/Counter0 external input)
PD5	T1 (Timer/Counter1 external input)
PD6	ICP (Timer/Counter1Input Capture pin)

When the pins are used for the alternate function the DDRD and PORTD register has to be set according to the alternate function description.

THE PORT D DATA REGISTER - PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D DATA DIRECTION REGISTER - DDRB

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	-	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D INPUT PINS ADDRESS

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port D Input Pins address - PIND - is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

PORTD AS GENERAL DIGITAL I/O

PDn, General I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PORTDn is set (one) when configured as an input pin the MOS pull up resistor is activated. To switch the pull up resistor off the PORTDn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 19. DDDn Bits on Port D Pins

DDDn	PORTDn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PDn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 6…0, pin number.

ALTERNATE FUNCTIONS OF PORTD

The alternate functions of Port D are:

<u>ICP - PORTD, Bit 6:</u>
Timer/Counter1 Input Capture pin. See the Timer/Counter1 description for further details.

<u>T1 - PORTD, Bit 5:</u>
T1, Timer 1 clock source. See the timer description for further details.

<u>T0 - PORTD, Bit 4:</u>
T0, Timer/Counter0 clock source. See the Timer description for further details.

<u>INT1 - PORTD, Bit 3:</u>
INT1, External Interrupt source 1. The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

INT0 - PORTD, Bit 2:
INT0, External Interrupt source 0. The PD2 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

TXD - PORTD, Bit 1:
Transmit Data (Data output pin for the UART). When the UART transmitter is enabled, this pin is configured as an output regardless of the value of DDRD1.

RXD - PORTD, Bit 0:
Receive Data (Data input pin for the UART). When the UART receiver is enabled this pin is configured as an output regardless of the value of DDRD0. When the UART forces this pin to be an input, a logical one in PORTD0 vill turn on the internal pull-up.

PORTD SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

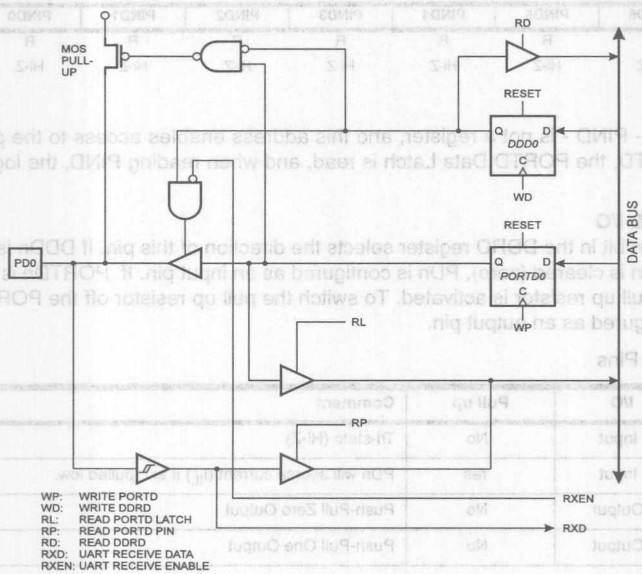


Figure 45. PORTD Schematic Diagram (Pin PD0)

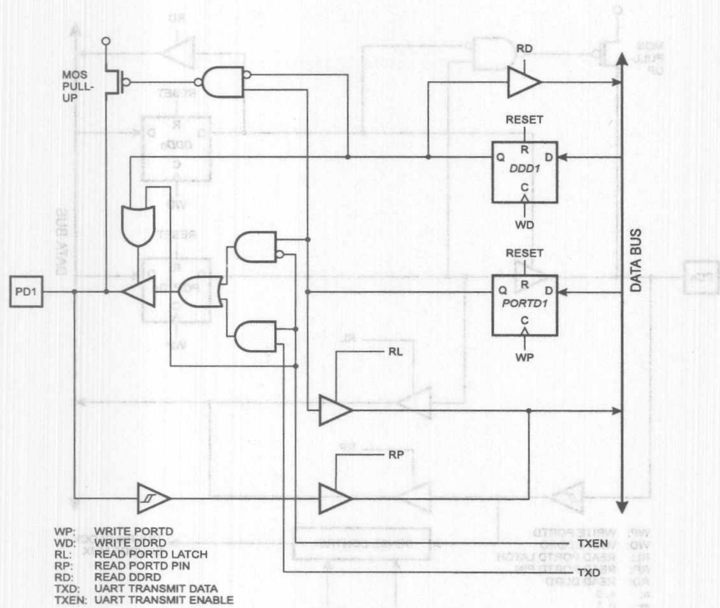


Figure 46. PORTD Schematic Diagram, Pin PD1

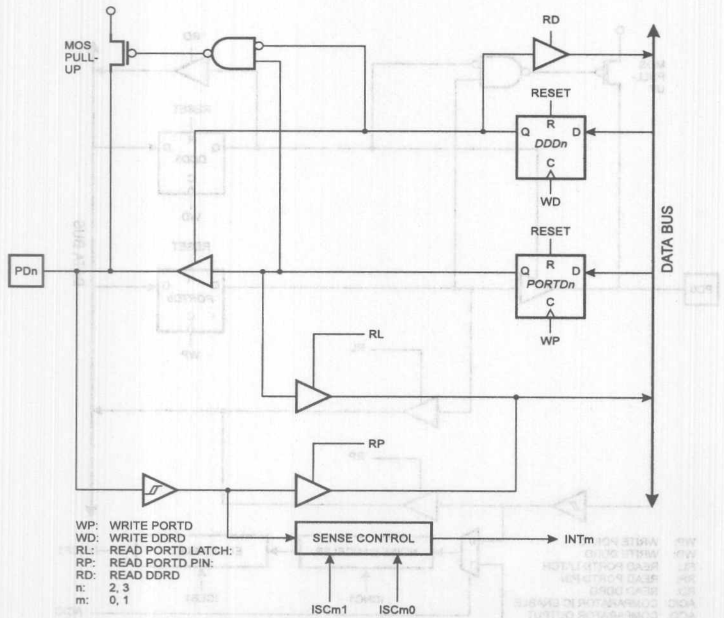


Figure 47. PORTD Schematic Diagram (Pins PD2 and PD3)

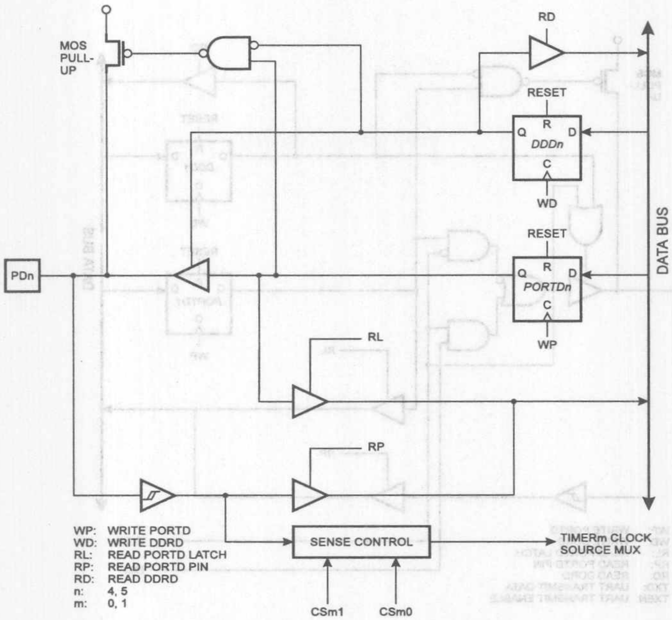


Figure 48. PORTD Schematic Diagram (Pins PD4 and PD5)

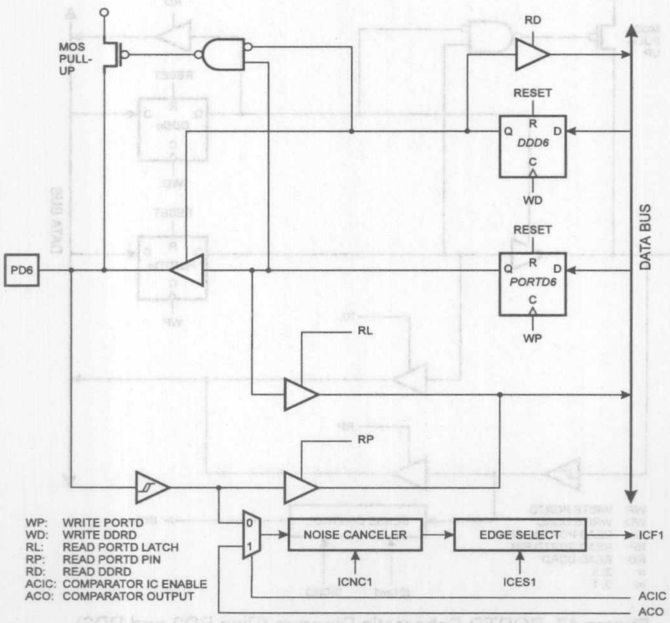


Figure 49. PORTD Schematic Diagram (Pin PD6)

Memory Programming

Program Memory Lock Bits

The AT90S2313 MCU provides two lock bits which can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table 20.

Table 20. Lock Bit Protection Modes

Program Lock Bits			Protection Type
Mode	LB1	LB2	
1	1	1	No program lock features
2	0	1	Further programming of the Flash is disabled
3	0	0	Same as mode 2, but verify is also disabled.

Note: The Lock Bits can only be erased with the Chip Erase operation.

Fuse Bits

The AT90S2313 has two fuse bits, SPIEN and FSTRT.

- When SPIEN is programmed ('0'), Serial Program Downloading is enabled. Default value is programmed ('0').
- When FSTRT is programmed ('0'), the short start-up time is selected. Default value is unprogrammed ('1'). Parts with this bit pre-programmed ('0') can be delivered on demand. See

These bits are not accessible in Serial Programming Mode and are not affected by a chip erase.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode. The three bytes reside in a separate address space, and for the AT90S2313 they are:

1. $000: $1E (indicates manufactured by Atmel)
2. $001: $91 (indicates 2 kB Flash memory)
3. $002: $01 (indicates 90S2313 device when $001 is $90)

Programming the Flash and EEPROM

Atmel's AT90S2313 offers 2K bytes of in-system reprogrammable Flash Program memory and 128 bytes of EEPROM Data memory.

The AT90S2313 is normally shipped with the on-chip Flash Program and EEPROM Data memory arrays in the erased state (i.e. contents = $FF) and ready to be programmed. This device supports a High-Voltage (12V) Parallel programming mode and a Low-Voltage Serial programming mode. The +12V is used for programming enable only, and no current of significance is drawn by this pin. The serial programming mode provides a convenient way to download the Program and Data into the AT90S2313 inside the user's system.

The Program and EEPROM memory arrays in the AT90S2313 are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided with the self-timed programming operation in the serial programming mode.

Parallel Programming

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory + Program Memory Lock bits and Fuse bits in the AT90S2313.

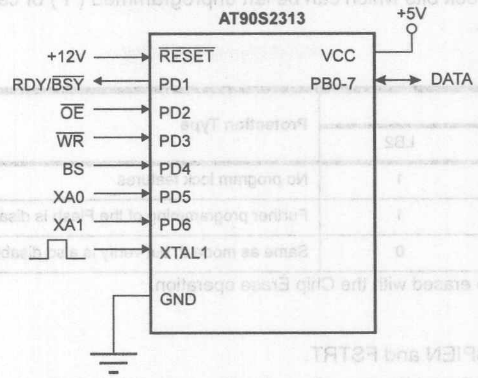


Figure 50. Parallel Programming

SIGNAL NAMES

In this section, some pins of the AT90S2313 are referenced by signal names describing their functionality during parallel programming rather than their pin names. Pins not described in the following table are referenced by pin names.

Table 21. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY / $\overline{\text{BSY}}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{\text{OE}}$	PD2	I	Output Enable (Active Low)
$\overline{\text{WR}}$	PD3	I	Write Pulse (Active Low)
BS	PD4	I	Byte Select
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1

The XA1/XA0 bits determine the action taken when the XTAL1 pin is given a positive pulse. The bit settings are shown in the following table:

Table 22. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or Low address byte for Flash determined by BS)
0	1	Load Data (High or Low data byte for Flash determined by BS)
1	0	Load Command
1	1	No Action, Idle

When pulsing $\overline{\text{WR}}$ or $\overline{\text{OE}}$, the command loaded determines the action on input or output. The command is a byte where the different bits are assigned functions as shown in the following table:

Table 23. Command Byte Bit Coding

Bit#	Meaning when Set
7	Chip Erase
6	Write Fuse Bits. Located in the data byte at the following bit positions: D5: SPIEN Fuse, D0: FSTRT Fuse (Note: Write '0' to program, '1' to erase)
5	Write Lock Bits. Located in the data byte at the following bit positions: D1: LB1, D0: LB2 (Note: write '0' to program)
4	Write Flash or EEPROM (determined by bit 0)
3	Read signature row
2	Read Lock and Fuse Bits. Located in the data byte at the following bits positions: D7: LB1, D6: LB2, D5: SPIEN Fuse, D0: FSTRT Fuse (Note: '0' means programmed)
1	Read from Flash or EEPROM (determined by bit 0)
0	0 : Flash Access, 1 : EEPROM Access

ENTER PROGRAMMING MODE

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5 V between VCC and GND.
2. Set $\overline{\text{RESET}}$ and BS pins to '0' and wait at least 100 ns.
3. Apply 12V to $\overline{\text{RESET}}$ and wait at least 100 ns before changing BS.

CHIP ERASE

The chip erase will erase the Flash and EEPROM memories plus Lock bits. The lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A chip erase must be performed before the chip is programmed.

Load Command "Chip Erase"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'.
3. Set PB(7:0) to '1000 0000'. This is the command for Chip erase.
4. Give XTAL1 a positive pulse. This loads the command, and starts the erase of the Flash and EEPROM arrays. After pulsing XTAL1, give $\overline{\text{WR}}$ a negative pulse to enable lock bit erase at the end of the erase cycle, then wait for at least 10 ms. Chip erase does not generate any activity on the RDY/$\overline{\text{BSY}}$ pin.

PROGRAMMING THE FLASH

Load Command "Program Flash"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'
3. Set PB(7:0) to '0001 0000'. This is the command for Flash programming.
4. Give XTAL1 a positive pulse. This loads the command.

Load Address Low byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '0'. This selects Low address.
3. Set PB(7:0) = Address Low byte ($00 - $FF)
4. Give XTAL1 a positive pulse. This loads the Address Low byte.

Load Address High byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '1'. This selects High address.
3. Set PB(7:0) = Address High byte ($00 - $03)
4. Give XTAL1 a positive pulse. This loads the Address High byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data Low byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data Low byte

1. Set BS to ('0').
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high to program the next byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data High byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data High byte

1. Set BS to '1'.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY / $\overline{BSY}$ goes low.
3. Wait until RDY / $\overline{BSY}$ goes high to program the next byte.

The loaded command and address are retained in the device during programming. To simplify programming, the following should be considered.

- The command for Flash programming needs only be loaded before programming of the first byte.
- Address High byte needs only be loaded before programming a new 256 word page in the Flash.

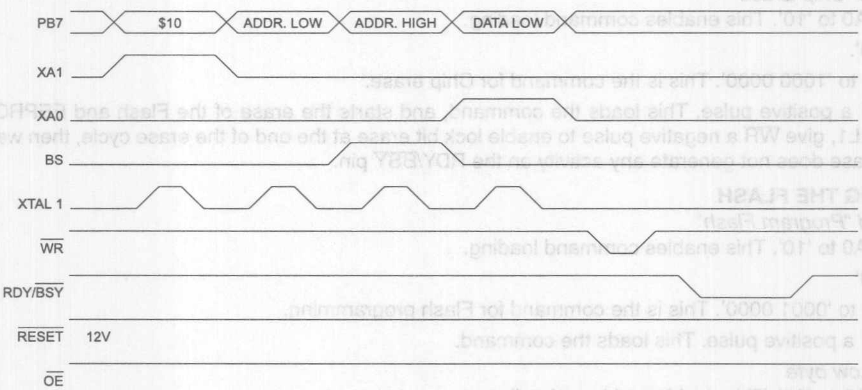


Figure 51. Programming Flash Low Byte

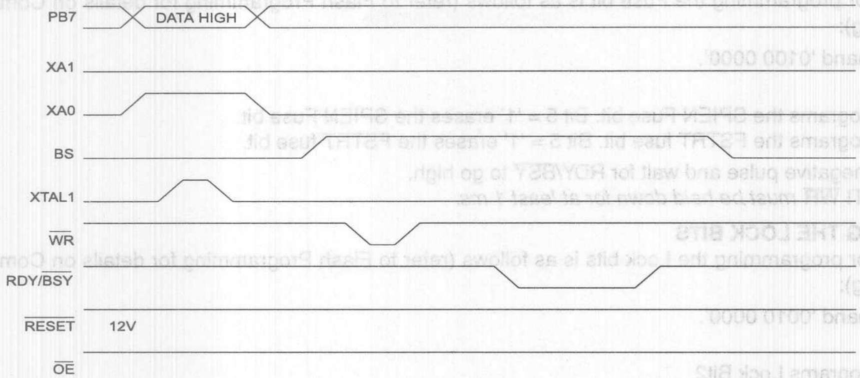


Figure 52. Programming Flash High Byte

PROGRAMMING THE EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0001 0001'.
2. Load EEPROM Address ($00 - $7F)
3. Load EEPROM Data ($00 - $FF)
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The Command needs only be loaded before programming the first byte.

READING THE FLASH

The algorithm for reading the Flash memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0010'.
2. Load Low Address ($00 - $FF)
3. Load High Address ($00 - $03)
4. Set $\overline{OE}$ to '0', and BS to '0'. The Low Data byte can now be read at PB(7:0)
5. Set BS to '1'. The High Data byte can now be read from PB(7:0)
6. Set $\overline{OE}$ to '1'.

The Command needs only be loaded before reading the first byte.

READING THE EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0011'.
2. Load EEPROM Address ($00 - $7F)
3. Set $\overline{OE}$ to '0', and BS to '0'. The EEPROM Data byte can now be read at PB(7:0)
4. Set $\overline{OE}$ to '1'.

The Command needs only be loaded before reading the first byte.

PROGRAMMING THE FUSE BITS

The algorithm for programming the Fuse bit is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0100 0000'.
2. Load Data.
   Bit 5 = '0' programs the SPIEN Fuse bit. Bit 5 = '1' erases the SPIEN Fuse bit.
   Bit 0 = '0' programs the FSTRT fuse bit. Bit 5 = '1' erases the FSTRT fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.
   IMPORTANT! $\overline{WR}$ *must be held down for at least 1 ms.*

PROGRAMMING THE LOCK BITS

The algorithm for programming the Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0010 0000'.
2. Load Data.
   Bit 2 = '0' programs Lock Bit2
   Bit 1 = '0' programs Lock Bit1
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The lock bits can only be cleared by executing a chip erase.

READING THE FUSE AND LOCK BITS

The algorithm for reading the Fuse and Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0100'.
2. Set $\overline{OE}$ to '0', and BS to '1'. The Status of Fuse and Lock bits can now be read at PB(7:0)
   Bit 7: Lock Bit1 ('0' means programmed)
   Bit 6: Lock Bit2 ('0' means programmed)
   Bit 5: SPIEN Fuse ('0' means programmed, '1' means erased)
   Bit 0: FSTRT Fuse ('0' means programmed, '1' means erased)
3. Set $\overline{OE}$ to '1'.

Observe especially that BS needs to be set to '1'.

READING THE SIGNATURE BYTES

The algorithm for reading the Signature Bytes bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 1000'.
2. Load Low address ($00 - $02)
   Set $\overline{OE}$ to '0', and BS to '0'. The Selected Signature byte can now be read at PB(7:0)
3. Set $\overline{OE}$ to '1'.

The command needs only be programmed before reading the first byte.

Serial Downloading

Both the Program and Data memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into $FF.

The Program and EEPROM memory arrays have separate address spaces, $000 to $7FF for Program Flash memory and $000 to $07F for EEPROM Data memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 1 XTAL1 clock cycle

High: > 4 XTAL1 clock cycles

SERIAL PROGRAMMING ALGORITHM

To program and verify the AT90S2313 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 24):

1. *Power-up sequence:*

   Apply power between VCC and GND while $\overline{\text{RESET}}$ and SCK are set to '0'. (If the programmer can not guarantee that SCK is held low during power-up, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to '0'.) If a crystal is not connected across pins XTAL1 and XTAL2, apply a 0 to 20 MHz clock to the XTAL1 pin.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB5. Refer to the above section for minimum high and low periods for the serial clock input (SCK).
3. If a chip erase is performed (must be done to erase the Flash), wait 10ms, give $\overline{\text{RESET}}$ a positive pulse and start over again from Step 2.
4. The Flash or EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. The next byte can be written after 4 ms.
5. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.
6. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
7. Power-off sequence (if needed):

   Set XTAL1 to '0' (if a crystal is not used).

   Set $\overline{\text{RESET}}$ to '1'.

   Turn V_{CC} power off..

Table 24. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both 2K & 128 byte memory arrays
Read Program Memory	0010 **H**000	xxxx xx**aa**	**bbbb bbbb**	**oooo oooo**	Read **H**(high or low) data **o** from Program memory at word address **a:b**
Write Program Memory	0100 **H**000	xxxx xx**aa**	**bbbb bbbb**	**iiii iiii**	Write **H**(high or low) data **i** to Program memory at word address **a:b**
Read EEPROM Memory	1010 0000	xxxx xxxx	x**bbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**
Write EEPROM Memory	1100 0000	xxxx xxxx	x**bbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**
Write Lock Bits	1010 1100	111x x***21***x	xxxx xxxx	xxxx xxxx	Write lock bits. Set bits ***1,2***='0' to program lock bits.
Read Device Code	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Device Code **o** from address **b**

Notes: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1 - High Byte
o = data out
i = data in
x = don't care
1 = lock bit 1
2 = lock bit 2

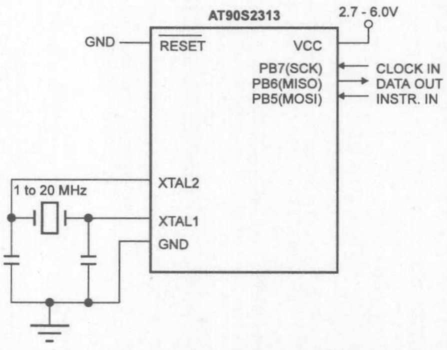


Figure 53. Serial Programming and Verify

When *writing* serial data to the AT90S2313, data is clocked on the *rising* edge of CLK.

When *reading* data from the AT90S2313, data is clocked on the *falling* edge of CLK. See Figure 54 for an explanation.

Programming Characteristics

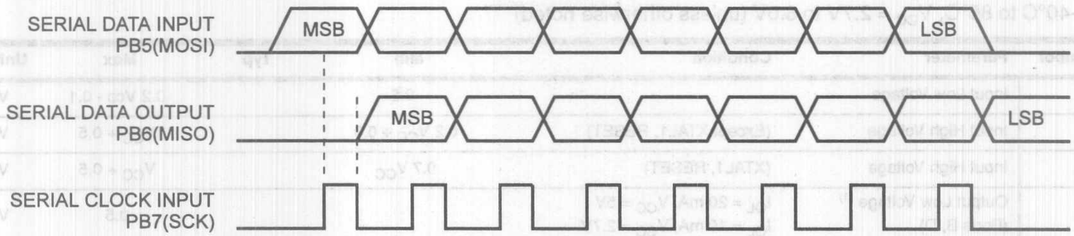


Figure 54. Serial Downloading Waveforms

Absolute Maximum Ratings*

Operating Temperature	-40°C to +105°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
I/O Pin Maximum Current	40.0 mA
Maximum Current VCC and GND	140.0 mA

*NOTICE: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

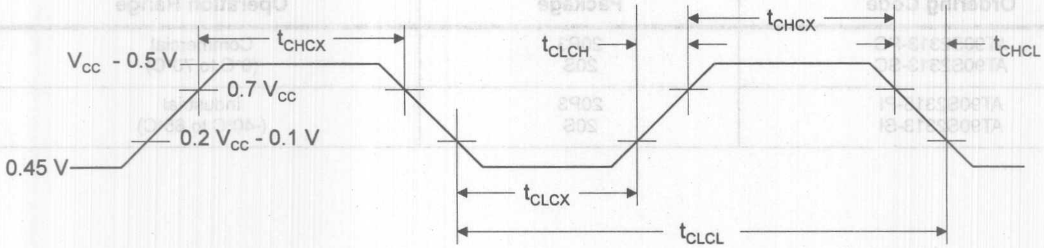
DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		0.2 Vcc - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RESET)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RESET)	0.7 V_{CC}		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage [1] (Ports B, D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V			0.5	V
V_{OH}	Output High Voltage (Ports B,D)	I_{HI} = 10 mA, V_{CC} = 5V I_{HI} = 5 mA, V_{CC} = 2.7V	4.5			V
I_{OH}	Output Source Current (Ports B,D)	V_{CC} = 5V V_{CC} = 2.7V			10 5	mA
I_{OL}	Output Sink Current (Port B,D)	V_{CC} = 5V V_{CC} = 2.7V			20 10	mA
RRST	Reset Pull-Up Resistor		100		500	kΩ
$R_{I/O}$	I/O Pin Pull-Up Resistor		35		120	kΩ
I_{CC}	Power Supply Current	Active Mode, 3V, 4MHz		2.5		mA
		Idle Mode 3V, 4MHz		800		uA
I_{CC}	Power Down Mode [2]	WDT enabled, 3V		50		µA
		WDT disabled, 3V		<1		µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V			20	mV
I_{ACLK}	Analog Comparator Input Leakage Current		1	5	10	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

Notes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 20mA
   Maximum total I_{OL} for all output pins: 80mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification.
   Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	4	0	16	MHz
t_{CLCL}	Clock Period	250		62.5		ns
t_{CHCX}	High Time	40		16.7		ns
t_{CLCX}	Low Time	40		16.7		ns
t_{CLCH}	Rise Time		10		4.15	ns
t_{CHCL}	Fall Time		10		4.15	ns

Ordering Information

Ordering Code	Package	Operation Range
AT90S2313-PC AT90S2313-SC	20P3 20S	Commercial (0°C to 70°C)
AT90S2313-PI AT90S2313-SI	20P3 20S	Industrial (-40°C to 85°C)

Package Type	
20P3	20 Lead, 0.300" Wide, Plastic Dual In-Line Package (PDIP)
20S	20 Lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)

AT90S2313 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	3-21
$3E ($5E)	Reserved									
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	3-22
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-	3-27
$3A ($5A)	GIFR	INTF1	INTF0							3-27
$39 ($59)	TIMSK	TOIE1	OCIE1A	-	-	TICIE1	-	TOIE0	-	3-27
$38 ($58)	TIFR	TOV1	OCF1A	-	-	ICF1	-	TOV0	-	3-28
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	-	-	SE	SM	ISC11	ISC10	ISC01	ISC00	3-29
$34 ($54)	Reserved									
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	3-32
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								3-33
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	TCCR1A	COM1A1	COM1A0	-	-	-	-	PWM11	PWM10	3-35
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	3-35
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte								3-36
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								3-36
$2B ($4B)	OCR1AH	Timer/Counter1 - Compare Register High Byte								3-37
$2A ($4A)	OCR1AL	Timer/Counter1 - Compare Register Low Byte								3-37
$29 ($49)	Reserved									
$28 ($48)	Reserved									
$27 ($47)	Reserved									
$26 ($46)	Reserved									
$25 ($45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								3-37
$24 ($44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								3-37
$23 ($43)	Reserved									
$22 ($42)	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	3-39
$20 ($40)	Reserved									
$1F ($3F)	Reserved									
$1E ($3E)	EEAR	-	EEPROM Address Register							3-40
$1D ($3D)	EEDR	EEPROM Data register								3-41
$1C ($3C)	EECR	-	-	-	-	-	EEMWE	EEWE	EERE	3-41
$1B ($3B)	Reserved									
$1A ($3A)	Reserved									
$19 ($39)	Reserved									
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	3-50
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	3-50
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	3-50
$15 ($35)	Reserved									
$14 ($34)	Reserved									
$13 ($33)	Reserved									
$12 ($32)	PORTD	-	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	3-55
$11 ($31)	DDRD	-	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	3-55
$10 ($30)	PIND	-	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	3-55
$0F ($2F)	Reserved									
$0E ($2E)	Reserved									
$0D ($2D)	Reserved									
$0C ($2C)	UDR	UART I/O Data Register								3-44
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-	3-44
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	3-45
$09 ($29)	UBRR	UART Baud Rate Register								3-47
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	3-48
…	Reserved									
$00 ($20)	Reserved									

AT90S2313 Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd – Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd – K	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl – K	Z,C,N,V,S	2
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd – Rr – C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd – K – C	Z,C,N,V,H	1
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF – K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBIS	P, b	Skip if Bit in I/O Register is Set	if (R(b)=1) PC ← PC + 2 or 3	None	1 / 2
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC + k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC + k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

(continued)

Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	3
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	3
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

Contents

Features

- Utilizes the *AVR* ® Enhanced RISC Architecture
- AV*R* - High Performance and Low Power RISC Architecture
- 120 Powerful Instructions - Most Single Clock Cycle Execution
- 4K bytes of In-System Reprogrammable Downloadable Flash
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 256 bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 256 bytes Internal SRAM
- 32 x 8 General Purpose Working Registers
- 32 Programmable I/O Lines
- Programmable Serial UART
- SPI Serial Interface
- V_{CC}: 2.7 - 6.0V
- Fully Static Operation, 0 - 20 MHz
- Instruction Cycle Time: 50 ns @ 20 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- One 16-Bit Timer/Counter with Separate Prescaler and Compare and Capture Modes
- Dual PWM
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes
- Programming Lock for Software Security

8-Bit AVR® Microcontroller with 4K bytes Downloadable Flash

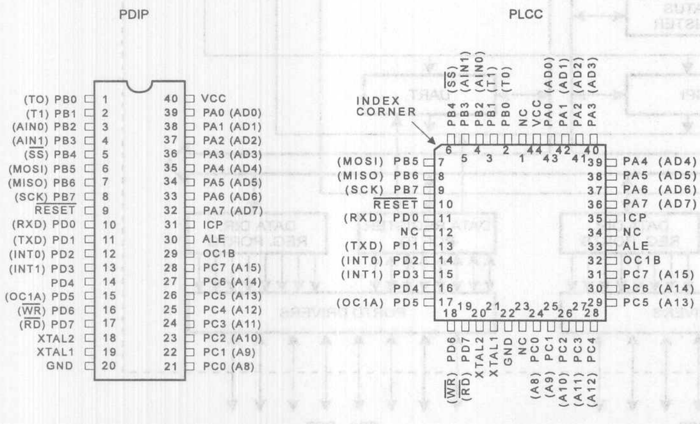
Preliminary

4

Description

The AT90S4414 is a low-power CMOS 8-bit microcontroller based on the *AVR* ® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S4414 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Pin Configurations



0840A

Block Diagram

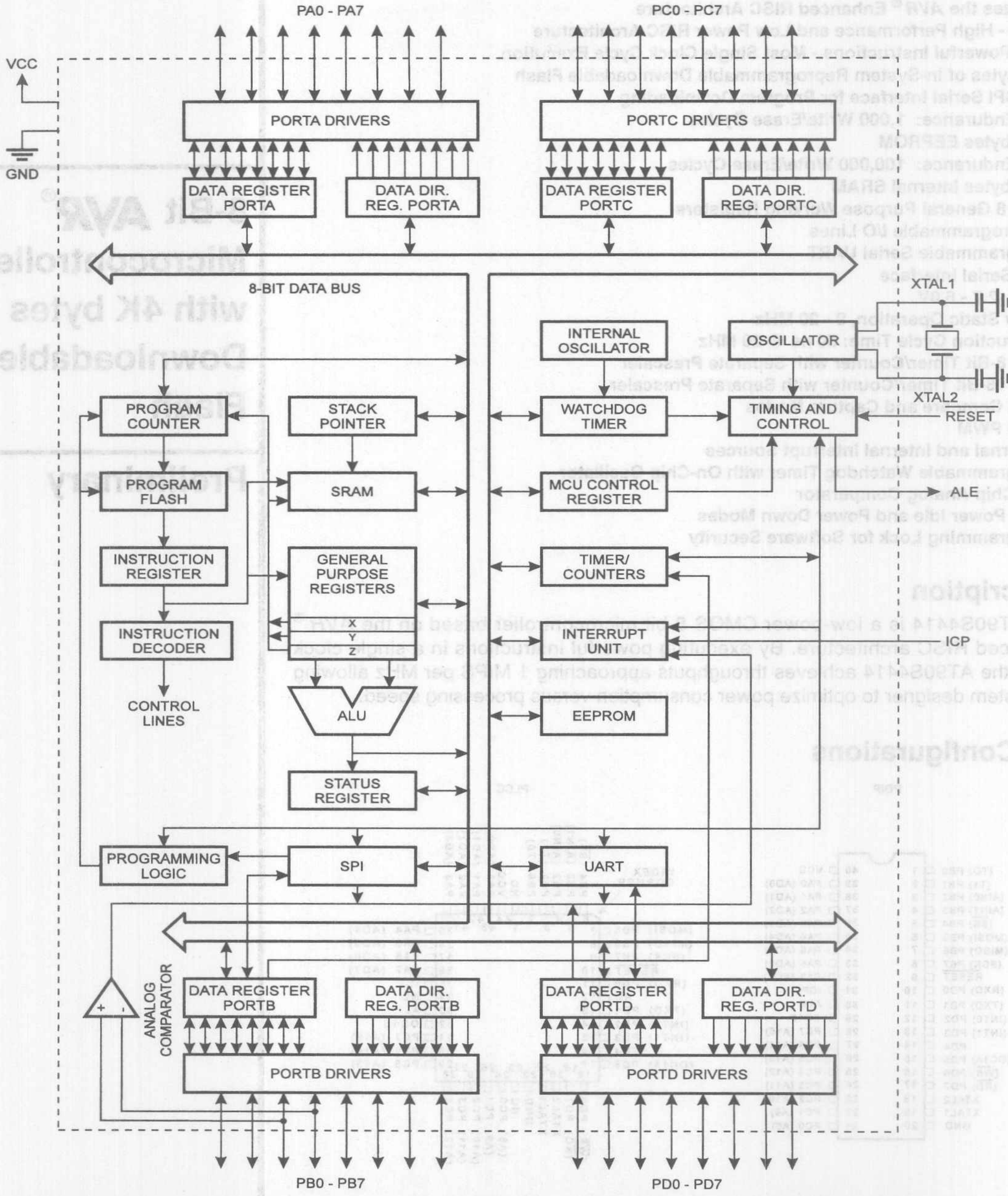


Figure 1. The AT90S4414 Block Diagram

Description (Continued)

The *AVR* core is based on an enhanced RISC architecture that combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The AT90S4414 provides the following features: 4K bytes of Downloadable Flash, 256 bytes EEPROM, 256 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, flexible timer/counters with compare modes, internal and external interrupts, a programmable serial UART, programmable Watchdog Timer with internal oscillator, an SPI serial port and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip Downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining an enhanced RISC 8-bit CPU with Downloadable Flash on a monolithic chip, the Atmel AT90S4414 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90S4414 *AVR* is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Supply voltage

GND

Ground

Port A (PA7..PA0)

Port A is an 8-bit bidirectional I/O port. Port pins can provide internal pullups (selected for each bit). The Port A output buffers can sink 20mA and can drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

Port A serves as Multiplexed Address/Data input/output when using external SRAM.

Port B (PB7..PB0)

Port B is an 8-bit bidirectional I/O pins with internal pullups. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port B also serves the functions of various special features of the AT90S4414 as listed on Page 4-62.

Port C (PC7..PC0)

Port C is an 8-bit bidirectional I/O port with internal pullups. The Port C output buffers can sink 20 mA. As inputs, Port C pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port C also serves as Address output when using external SRAM.

Port D (PD7..PD0)

Port D is an 8-bit bidirectional I/O port with internal pullups. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port D also serves the functions of various special features of the AT90S4414 as listed on Page 4-68

RESET

Reset input. A low on this pin for two machine cycles while the oscillator is running resets the device.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

ICP

ICP is the input pin for the Timer/Counter1 Input Capture function.

OC1B

OC1B is the output pin for the Timer/Counter1 Output CompareB function

ALE

ALE is the Address Latch Enable used when the External Memory is enabled. The ALE strobe is used to latch the low-order address (8 bits) into an address latch during the first access cycle, and the AD0-7 pins are used for data during the second access cycle.

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

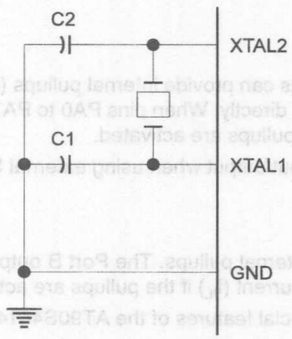


Figure 2. Oscillator Connections

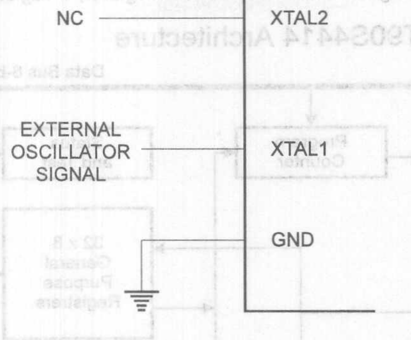


Figure 3. External Clock Drive Configuration

AT90S4414 *AVR* Enhanced RISC Microcontroller CPU

The AT90S4414 *AVR* RISC microcontroller is upward compatible with the *AVR* Enhanced RISC Architecture. The programs written for the AT90S4414 MCU are fully compatible with the range of *AVR* 8-bit MCUs (AT90Sxxxx) with respect to source code and clock cycles for execution.

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one ALU (Arithmetic Logic Unit) operation is executed. Two operands are output from the register file, the operation is executed, and the result is stored back in the register file - in one clock cycle.

Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing - enabling efficient address calculations. One of the three address pointers is also used as the address pointer for the constant table look up function. These added function registers are the 16-bits X-register, Y-register and Z-register.

AVR AT90S4414 Architecture

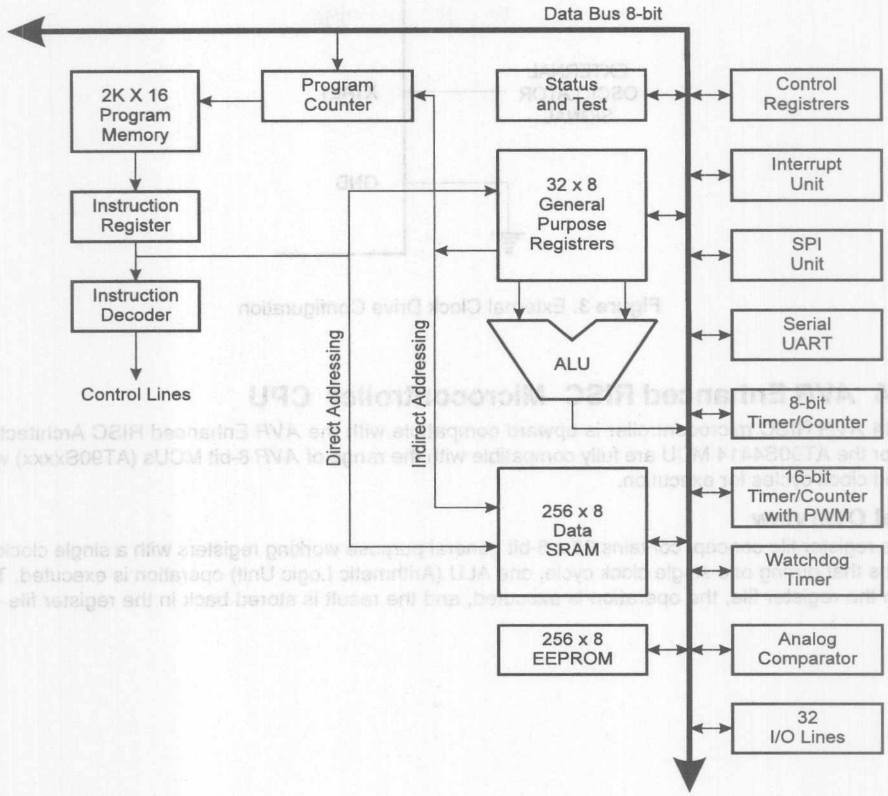


Figure 4. The AT90S4414 *AVR* Enhanced RISC Architecture

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S4414 *AVR* Enhanced RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the register file, $20 - $5F.

The *AVR* uses a Harvard architecture concept - with separate memories and buses for program and data. The program memory is executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system downloadable Flash memory.

With the relative jump and call instructions, the whole 2K address space is directly accessed. Most *AVR* instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 16-bit stack pointer SP is read/write accessible in the I/O space.

The 256 bytes data SRAM can be easily accessed through the five different addressing modes supported in the *AVR* architecture.

The memory spaces in the *AVR* architecture are all linear and regular memory maps.

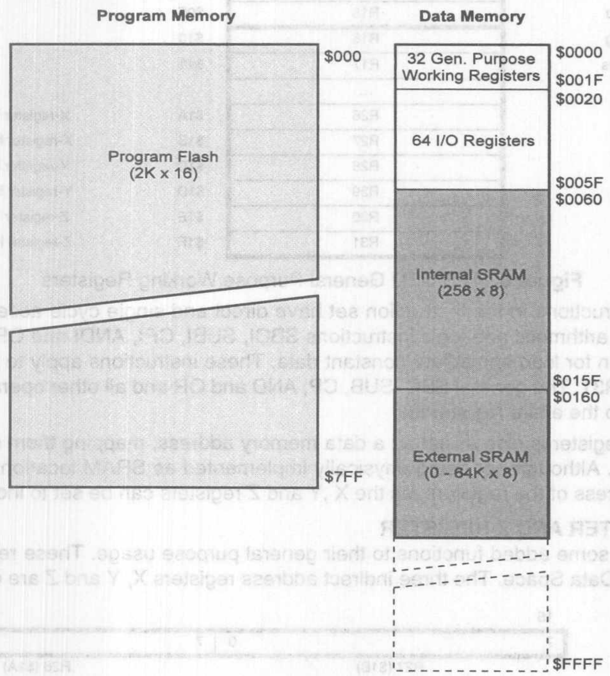


Figure 5. Memory Maps

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt address vector the higher priority.

The General Purpose Register File

Figure 6 shows the structure of the 32 general purpose working registers in the CPU.

	7 0	Addr.	
General Purpose Working Registers	R0	$00	
	R1	$01	
	R2	$02	
	…		
	R13	$0D	
	R14	$0E	
	R15	$0F	
	R16	$10	
	R17	$11	
	…		
	R26	$1A	X-register low byte
	R27	$1B	X-register high byte
	R28	$1C	Y-register low byte
	R29	$1D	Y-register high byte
	R30	$1E	Z-register low byte
	R31	$1F	Z-register high byte

Figure 6. *AVR* CPU General Purpose Working Registers

All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI and ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file - R16..R31. The general SBC, SUB, CP, AND and OR and all other operations between two registers or on a single register apply to the entire register file.

As shown in Figure 6, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X ,Y and Z registers can be set to index any register in the file.

THE X-REGISTER, Y-REGISTER AND Z-REGISTER

The registers R26..R31 have some added functions to their general purpose usage. These registers are address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y and Z are defined as:

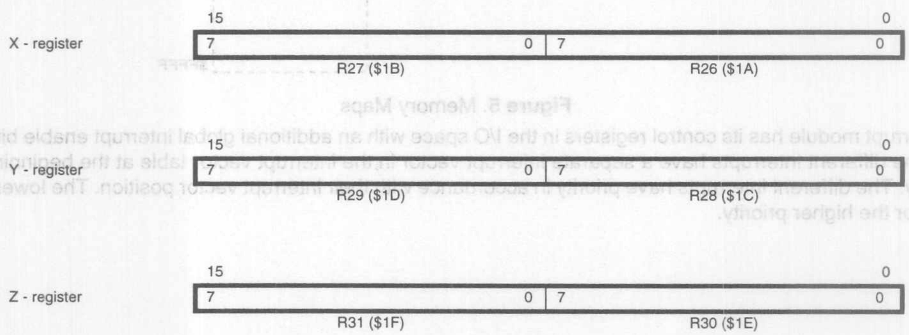


Figure 7. The X, Y and Z Registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

The ALU - Arithmetic Logic Unit

The high-performance *AVR* ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories - arithmetic, logic and bit-functions. Some microcontrollers in the *AVR* product family feature a hardware multiplier in the arithmetic part of the ALU.

The Downloadable Flash Program Memory

The AT90S4414 contains 4K bytes on-chip downloadable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 2K x 16. The Flash memory has an endurance of at least 1000 write/erase cycles. The AT90S4414 Program Counter (PC) is 11 bits wide, thus addressing the 2048 program memory addresses. See Page 4-78 for a detailed description on Flash data downloading.

Constant tables must be allocated within the address 0-2K (see the LPM - Load Program Memory instruction description).

See Page 4-14 for the different program memory addressing modes.

The SRAM Data Memory - Internal and External

The following figure shows how the AT90S4414 SRAM Memory is organized:

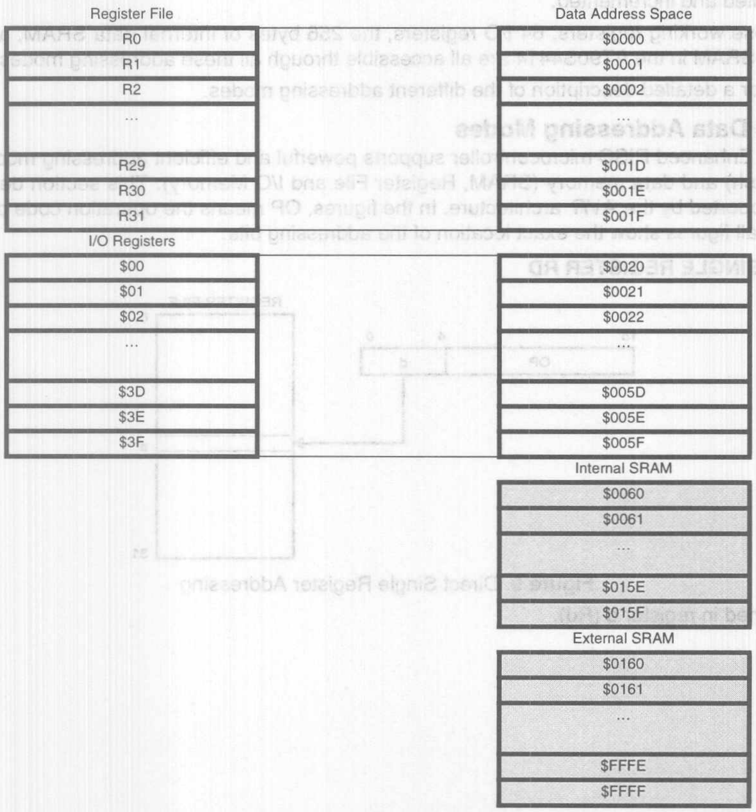


Figure 8. SRAM Organization

The lower 352 Data Memory locations address the Register file, the I/O Memory and the internal data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 256 locations address the internal data SRAM. An optional external data SRAM can be placed in the same SRAM memory space. This SRAM will occupy the location following the internal SRAM and up to as much as 64K - 1, depending on SRAM size.

When the addresses accessing the data memory space exceeds the internal data SRAM locations, the external data SRAM is accessed using the same instructions as for the internal data SRAM access. When the internal data space is accessed, the read and write strobe pins ($\overline{RD}$ and $\overline{WR}$) are inactive during the whole access cycle. External SRAM operation is enabled by setting the SRE bit in the MCUCR register. See Page 4-31 for details.

Accessing external SRAM takes one additional clock cycle per byte compared to the internal SRAM. This applies to commands LD, ST, LDS, STS, PUSH, POP. If the stack is placed in the external SRAM, interrupts, subroutine calls and returns will require two more clock cycles. When the external SRAM is used with wait state, all external SRAM access takes four clock cycles extra.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-Decrement and Indirect with Post-Increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode features a 63 address locations reach from the base address given by the Y or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y and Z are decremented and incremented.

The 32 general purpose working registers, 64 I/O registers, the 256 bytes of internal data SRAM, and the 64K bytes of optional external data SRAM in the AT90S4414 are all accessible through all these addressing modes.

See the next section for a detailed description of the different addressing modes.

The Program and Data Addressing Modes

The AT90S4414 *AVR* Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the program memory (Flash) and data memory (SRAM, Register File and I/O Memory). This section describes the different addressing modes supported by the *AVR* architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

REGISTER DIRECT, SINGLE REGISTER RD

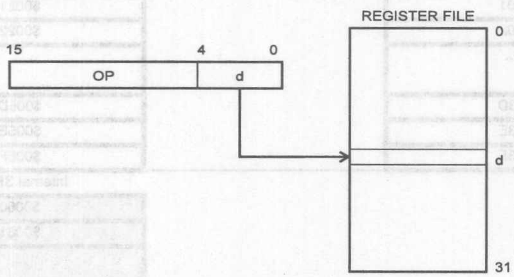


Figure 9. Direct Single Register Addressing

The operand is contained in register d (Rd).

REGISTER DIRECT, TWO REGISTERS RD AND RR

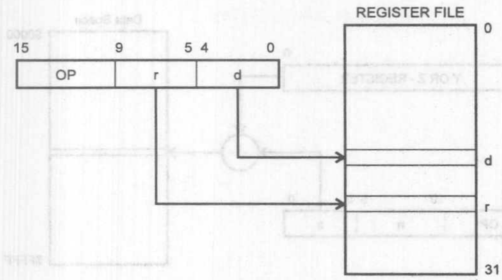


Figure 10. Direct Register Addressing, Two Registers

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O DIRECT

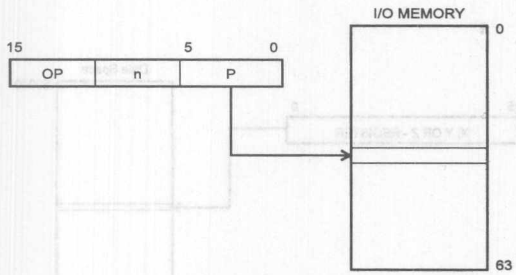


Figure 11. I/O Direct Addressing

Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

DATA DIRECT

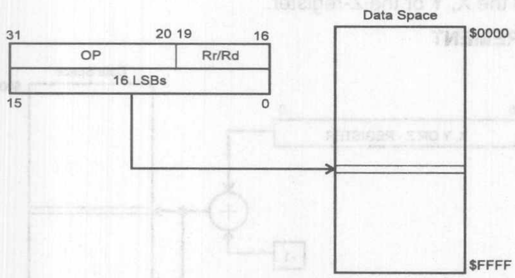


Figure 12. Direct Data Addressing

A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

DATA INDIRECT WITH DISPLACEMENT

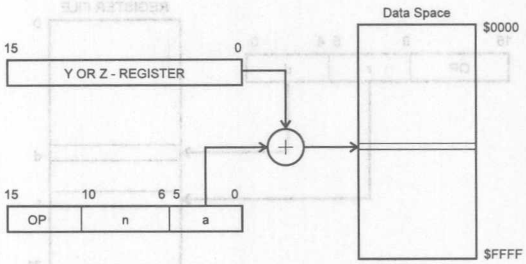


Figure 13. Data Indirect with Displacement

Operand address is the result of the Y or Z-register contents added to the address contained in 6 bits of the instruction word.

DATA INDIRECT

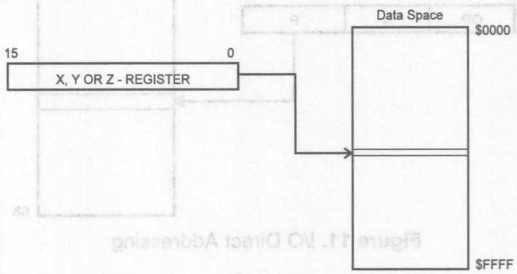


Figure 14. Data Indirect Addressing

Operand address is the contents of the X, Y or the Z-register.

DATA INDIRECT WITH PRE-DECREMENT

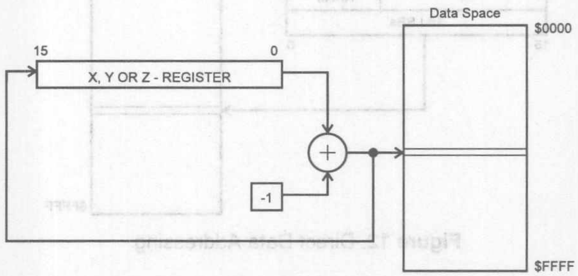


Figure 15. Data Indirect Addressing With Pre-Decrement

The X, Y or the Z-register is decremented before the operation. Operand address is the decremented contents of the X, Y or the Z-register.

DATA INDIRECT WITH POST-INCREMENT

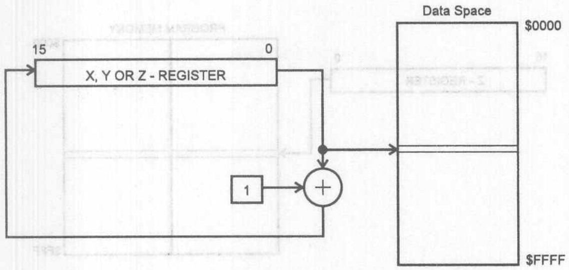


Figure 16. Data Indirect Addressing With Post-Increment

The X, Y or the Z-register is incremented after the operation. Operand address is the content of the X, Y or the Z-register prior to incrementing.

CONSTANT ADDRESSING USING THE LPM INSTRUCTION

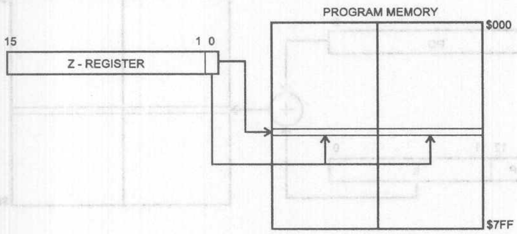


Figure 17. Code Memory Constant Addressing

Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 2K) and LSB, select low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

DIRECT PROGRAM ADDRESS, JMP AND CALL

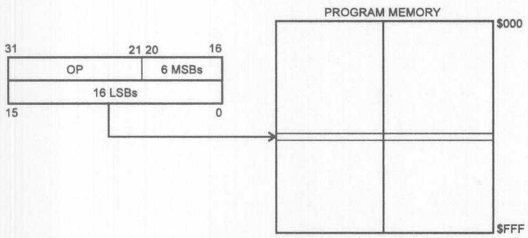


Figure 18. Direct Program Memory Addressing

Program execution continues at the address immediate in the instruction words.

INDIRECT PROGRAM ADDRESSING, IJMP AND ICALL

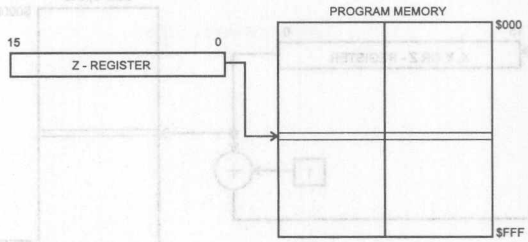


Figure 19. Indirect Program Memory Addressing

Program execution continues at address contained by the Z-register (i.e. the PC is loaded with the contents of the Z-register).

RELATIVE PROGRAM ADDRESSING, RJMP AND RCALL

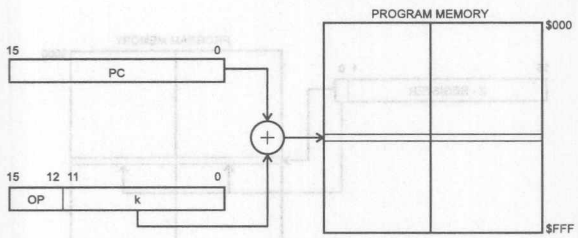


Figure 20. Relative Program Memory Addressing

Program execution continues at address PC + k. The relative address k is ± 2K.

The EEPROM Data Memory

The AT90S4414 contains 256 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on Page 4-44 specifying the EEPROM address register, the EEPROM data register, and the EEPROM control register.

For the SPI data downloading, see Page 4-78 for a detailed description.

Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The *AVR* CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 21 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

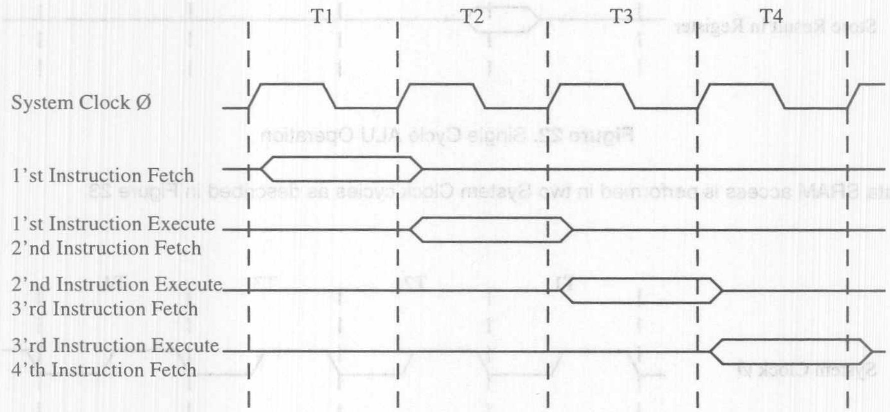


Figure 21. The Parallel Instruction Fetches and Instruction Executions

Figure 22 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

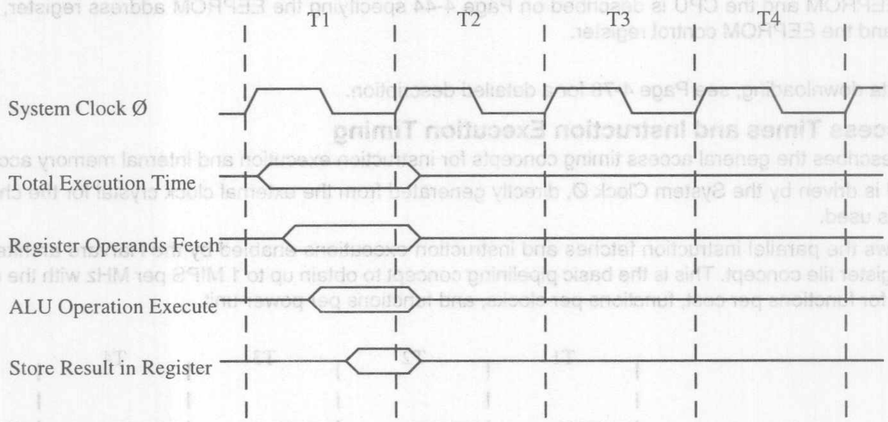


Figure 22. Single Cycle ALU Operation

The internal data SRAM access is performed in two System Clock cycles as described in Figure 23.

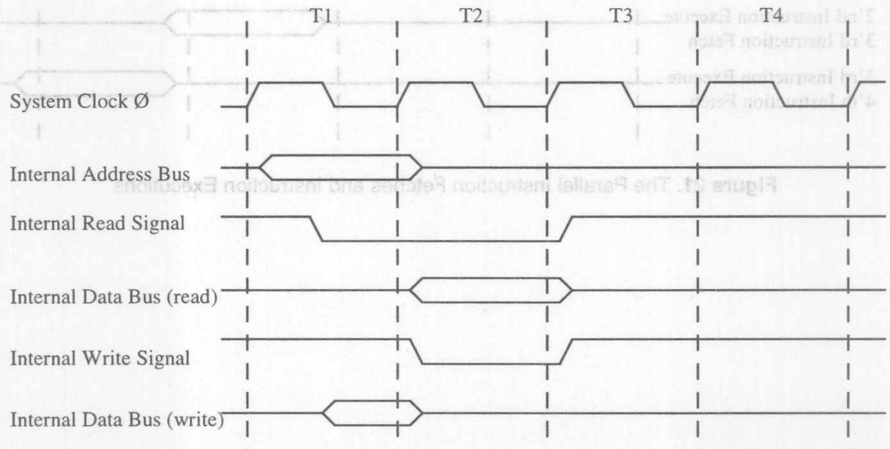


Figure 23. On-Chip Data SRAM Access Cycles

The external data SRAM access is performed in two System Clock cycles as described in Figure 23 .

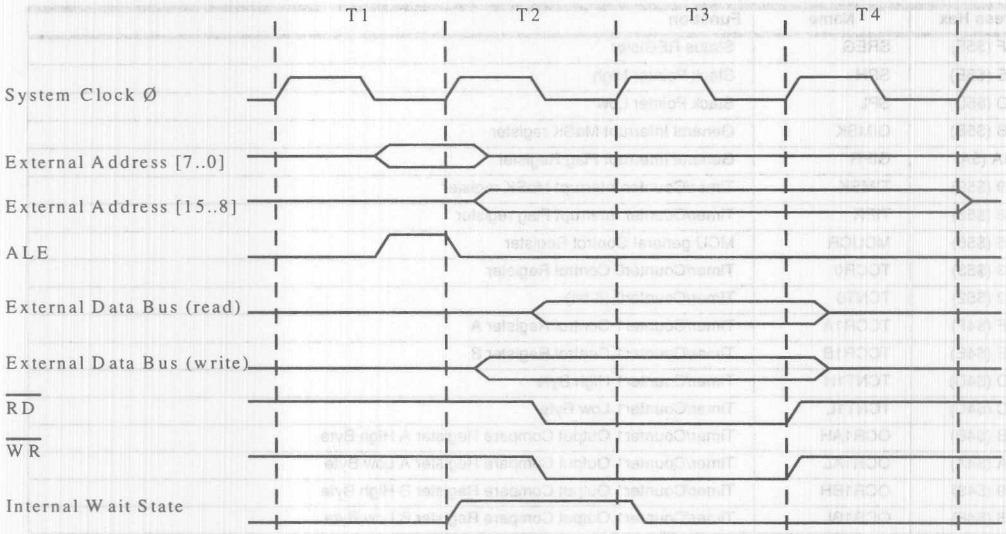


Figure 24. External Data SRAM Memory Cycles without Wait State

The external data SRAM memory access cycle with the Wait State bit enabled (Wait State active) is shown in Figure 25.

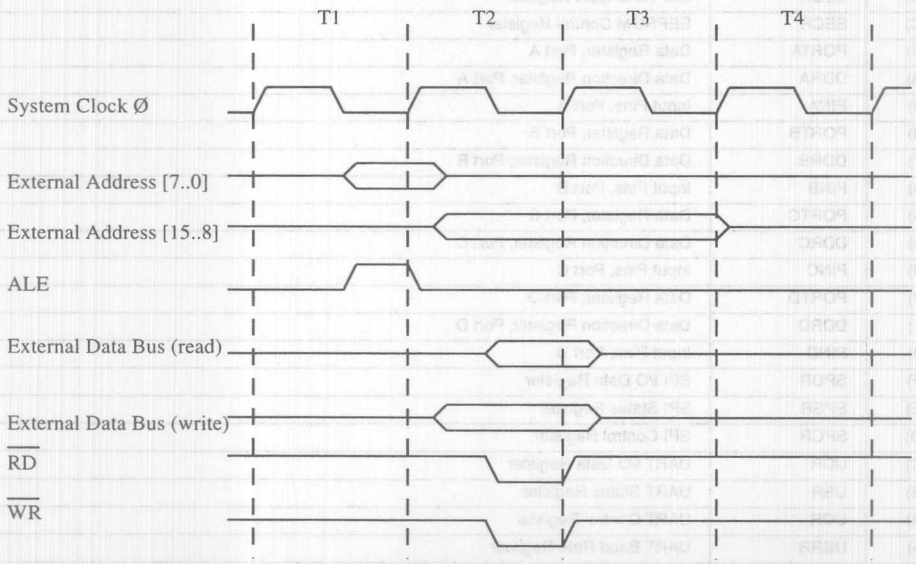


Figure 25. External Data SRAM Memory Cycles with Wait State

I/O Memory

The I/O space definition of the AT90S4414 is shown in the following table:

Table 1. AT90S4414 I/O Space

Address Hex	Name	Function
$3F ($5F)	SREG	Status REGister
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU general Control Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter1 Low Byte
$2B ($4B)	OCR1AH	Timer/Counter1 Output Compare Register A High Byte
$2A ($4A)	OCR1AL	Timer/Counter1 Output Compare Register A Low Byte
$29 ($49)	OCR1BH	Timer/Counter1 Output Compare Register B High Byte
$28 ($48)	OCR1BL	Timer/Counter1 Output Compare Register B Low Byte
$25 ($45)	ICR1H	T/C 1 Input Capture Register High Byte
$24 ($44)	ICR1L	T/C 1 Input Capture Register Low Byte
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1E ($3E)	EEAR	EEPROM Address Register
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$1B ($3B)	PORTA	Data Register, Port A
$1A ($3A)	DDRA	Data Direction Register, Port A
$19 ($39)	PINA	Input Pins, Port A
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$15 ($35)	PORTC	Data Register, Port C
$14 ($34)	DDRC	Data Direction Register, Port C
$13 ($33)	PINC	Input Pins, Port C
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	USR	UART Status Register
$0A ($2A)	UCR	UART Control Register
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register

Note: reserved and unused locations are not shown in the table

All the different AT90S4414 I/Os and peripherals are placed in the I/O space. The different I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set chapter for more details.

When using the I/O specific commands, IN, OUT,SBIS and SBIC, the I/O addresses $00 - $3F must be used. When addressing I/O registers as SRAM, $20 must be added to this address. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

The different I/O and peripherals control registers are explained in the following chapters.

THE STATUS REGISTER - SREG

The *AVR* status register - SREG - at I/O space location $3F ($5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - I : Global Interrupt Enable:

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in the interrupt mask registers - GIMSK and TIMSK. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled independent of the GIMSK and TIMSK values. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

Bit 6 - T : Bit Copy Storage:

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

Bit 5 - H : Half Carry Flag:

The half carry flag H indicates a half carry in some arithmetic operations. See the Instruction Set Description for detailed information.

Bit 4 - S : Sign Bit, $S = N \oplus V$:

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set Description for detailed information.

Bit 3 - V : Two's Complement Overflow Flag:

The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set Description for detailed information.

Bit 2 - N : Negative Flag:

The negative flag N indicates a negative result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 1 - Z : Zero Flag:

The zero flag Z indicates a zero result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 0 - C : Carry Flag:

The carry flag C indicates a carry in an arithmetic or logic operation. See the Instruction Set Description for detailed information.

THE STACK POINTER - SP

The general *AVR* 16-bit Stack Pointer is effectively built up of two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the AT90S4414 supports up to 64 kB external SRAM, all 16-bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when data is pushed onto the Stack with subroutine CALL and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt IRET.

Reset and Interrupt Handling

The AT90S4414 provides 12 different interrupt sources. These interrupts and the separate reset vector, each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 2. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 - the External Interrupt Request 0 etc.

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0, OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	UART, RX	UART, Rx Complete
11	$00A	UART, UDRE	UART Data Register Empty
12	$00B	UART, TX	UART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator

The most typical and general program setup for the Reset and Interrupt Vector Addresses are:

Address	Labels	Code		Comments
$000		rjmp	RESET	; Reset Handle
$001		rjmp	EXT_INT0	; IRQ0 Handle
$002		rjmp	EXT_INT1	; IRQ1 Handle
$003		rjmp	TIM1_CAPT	; Timer1 capture Handle
$004		rjmp	TIM1_COMPA	; Timer1 compareA Handle
$005		rjmp	TIM1_OVF	; Timer1 overflow Handle
$006		rjmp	TIM1_OVF	; Timer1 overflow Handle
$007		rjmp	TIM0_OVF	; Timer0 overflow Handle
$008		rjmp	SPI_HANDLE	; SPI TX Handle
$009		rjmp	UART_RXC	; UART RX Complete Handle
$00a		rjmp	UART_DRE	; UDR Empty Handle
$00b		rjmp	UART_TXC	; UART TX Complete Handle
$00c		rjmp	ANA_COMP	; Analog Comparator Handle
;				
$00d	MAIN:	<instr>	xxx	; Main program start
…	…	…		

RESET SOURCES

The AT90S4414 has three sources of reset:

- Power-On Reset. The MCU is reset when a supply voltage is applied to the VCC and GND pins.
- External Reset. The MCU is reset when a low level is present on the $\overline{\text{RESET}}$ pin for more than two XTAL cycles
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are then set to their initial values, and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP - relative jump - instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 26 shows the reset logic. Table 3 defines the timing and electrical parameters of the reset circuitry.

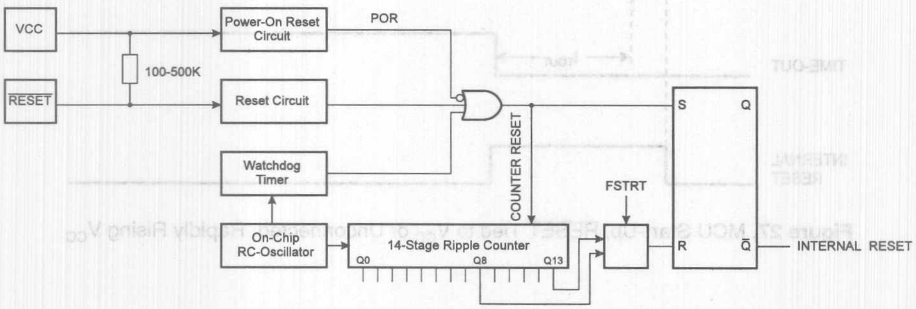


Figure 26. Reset Logic

Table 3. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-On Reset Threshold Voltage	1.8	2	2.2	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		$V_{CC}/2$		V
t_{POR}	Power-On Reset Period	2	3	4	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Unprogrammed	11	16	21	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Programmed	1.0	1.1	1.2	ms

POWER-ON RESET

A Power-On Reset (POR) circuit ensures that the device is not started until V_{CC} has reached a safe level. As shown in Figure 26, an internal timer clocked from the Watchdog timer oscillator prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-On Threshold voltage - V_{POT}, regardless of the V_{CC} rise time (see Figure 27 and Figure 28). The total reset period is the Power-On Reset period - t_{POR} + the Delay Time-out period - t_{TOUT}. The FSTRT fuse bit in the Flash can be programmed to give a shorter start-up time if a ceramic resonator or any other fast-start oscillator is used to clock the MCU.

As the pin is pulled high by an on-chip resistor, the pin can be left unconnected if no external reset is required. Connecting to V_{CC} will have the same effect. By holding the pin low for a period after V_{CC} has been applied, the Power-On Reset period can be extended. Refer to Figure 29 for a timing example on this.

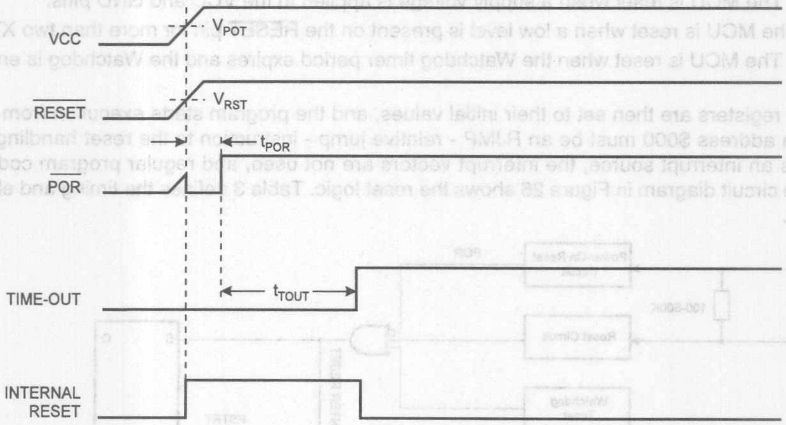


Figure 27. MCU Start-Up, $\overline{\text{RESET}}$ Tied to V_{CC} or Unconnected. Rapidly Rising V_{CC}

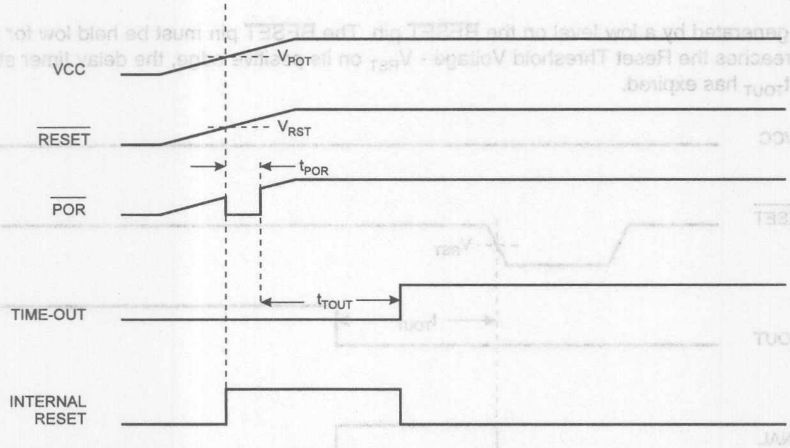


Figure 28. MCU Start-Up, RESET Tied to V_{CC} or Unconnected. Slowly Rising V_{CC}

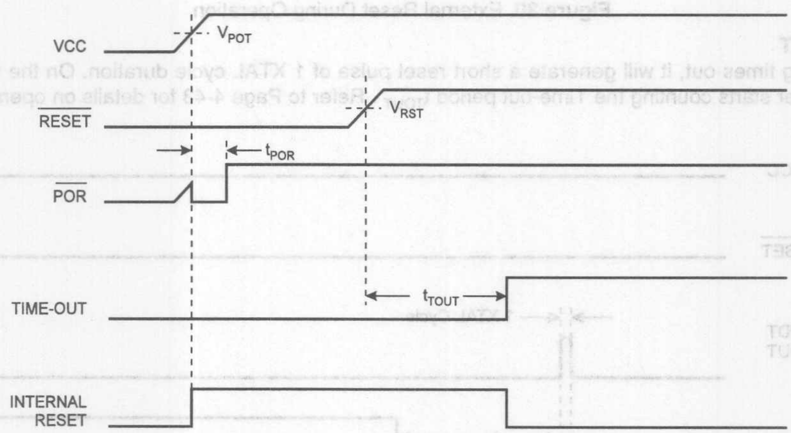


Figure 29. MCU Start-Up, RESET Controlled Externally

EXTERNAL RESET

An external reset is generated by a low level on the $\overline{RESET}$ pin. The $\overline{RESET}$ pin must be held low for at least two crystal clock cycles. When reaches the Reset Threshold Voltage - V_{RST} on its positive edge, the delay timer starts the MCU after the Time-out period t_{TOUT} has expired.

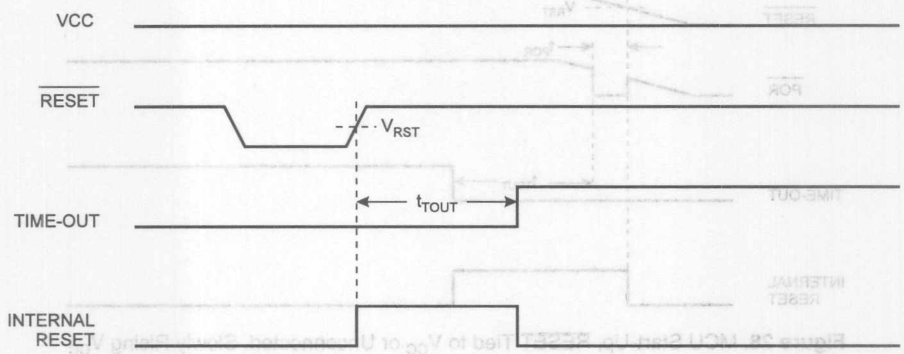


Figure 30. External Reset During Operation

WATCHDOG RESET

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to Page 4-43 for details on operation of the Watchdog.

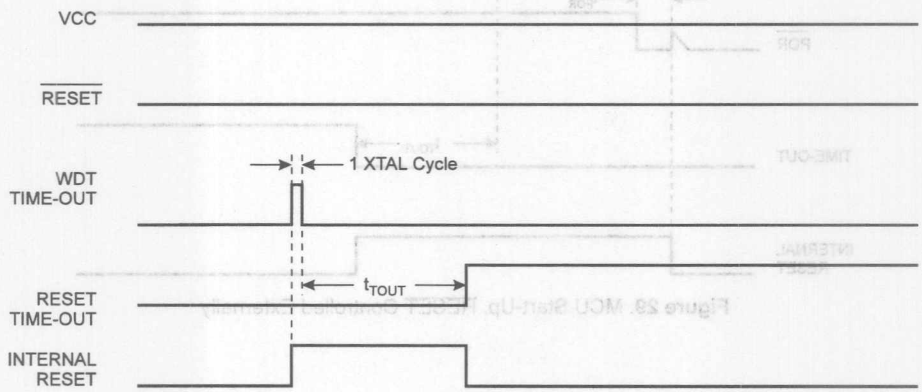


Figure 31. Watchdog Reset During Operation

INTERRUPT HANDLING

The AT90S4414 has two 8-bit Interrupt Mask control registers; GIMSK - General Interrupt Mask register and TIMSK - Timer/Counter Interrupt Mask register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software must set (one) the I-bit to enable interrupts.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

THE GENERAL INTERRUPT MASK REGISTER - GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INT1	INT0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INT1 : External Interrupt Request 1 Enable:
When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from program memory address $002. See also "external Interrupts".

Bit 6 - INT0 : External Interrupt Request 0 Enable:
When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from program memory address $001. See also "External Interrupts."

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and always read as zero.

THE GENERAL INTERRUPT FLAG REGISTER - GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INTF1 : External Interrupt Flag1:
When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $002. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bit 6 - INTF0 : External Interrupt Flag0:
When an event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $001. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and always read as zero.

THE TIMER/COUNTER INTERRUPT MASK REGISTER - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOIE1 : Timer/Counter1 Overflow Interrupt Enable:
When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt (at vector $006) is executed if an overflow in Timer/Counter1 occurs. The Overflow Flag (Timer/Counter1) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR. When Timer/Counter1 is in PWM mode, the Timer Overflow flag is set when the counter changes counting direction at $0000.

Bit 6 - OCE1A :Timer/Counter1 Output CompareA Match Interrupt Enable:
When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareA Match interrupt is enabled. The corresponding interrupt (at vector $004) is executed if a CompareA match in Timer/Counter1 occurs. The CompareA Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 5 - OCIE1B :Timer/Counter1 Output CompareB Match Interrupt Enable:
When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareB Match interrupt is enabled. The corresponding interrupt (at vector $005) is executed if a CompareB match in Timer/Counter1 occurs. The CompareB Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 4 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

Bit 3 - TICIE1 : Timer/Counter1 Input Capture Interrupt Enable:
When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Input Capture Event Interrupt is enabled. The corresponding interrupt (at vector $003) is executed if a capture-triggering event occurs on pin 31, ICP. The Input Capture Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

Bit 1 - TOIE0 : Timer/Counter0 Overflow Interrupt Enable:
When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt (at vector $008) is executed if an overflow in Timer/Counter0 occurs. The Overflow Flag (Timer0) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 0 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

THE TIMER/COUNTER INTERRUPT FLAG REGISTER - TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	TOV1	OCF1A	OCIFB	-	ICF1	-	TOV0	-	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOV1 : Timer/Counter1 Overflow Flag:
The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logic one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

Bit 6 - OCF1A : Output Compare Flag 1A:
The OCF1A bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1A - Output Compare Register 1A. OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare match InterruptA Enable), and the OCF1A are set (one), the Timer/Counter1 Compare match Interrupt is executed.

Bit 5 - OCF1B : Output Compare Flag 1B:
The OCF1B bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1B - Output Compare Register 1B. OCF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1 is cleared by writing a logic one to the flag.. When the I-bit in SREG, and OCIE1B (Timer/Counter1 Compare match InterruptB Enable), and the OCF1B are set (one), the Timer/Counter1 Compare match Interrupt is executed.

Bit 4 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

Bit 3 - ICF1 : - Input Capture Flag 1:
The ICF1 bit is set (one) to flag an input capture event, indicating that the Timer/Counter1 value has been transferred to the input capture register - ICR1. ICF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ICF1 is cleared by writing a logic one to the flag.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

Bit 1 - TOV0 : Timer/Counter0 Overflow Flag:
The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

Bit 0 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and always reads zero.

EXTERNAL INTERRUPTS

The external interrupts are triggered by the INT1 and INT0 pins.Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register - MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

The external interrupts are set up as described in the specification for the MCU Control Register - MCUCR.

INTERRUPT RESPONSE TIME

The interrupt execution response for all the enabled *AVR* interrupts is 4 clock cycles minimum. After the 4 clock cycles the program vector address for the actual interrupt handling routine is executed. During this 4 clock cycle period, the Program Counter (2 bytes) is pushed onto the Stack, and the Stack Pointer is decremented by 2. The vector is a relative jump to the interrupt routine, and this jump takes 2 clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine (same as for a subroutine call routine) takes 4 clock cycles. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, and the Stack Pointer is incremented by 2. When *AVR* exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register - SREG - is not handled by the *AVR* hardware, neither for interrupts nor for subroutines. For the interrupt handling routines requiring a storage of the SREG, this must be performed by user software.

For Interrupts triggered by events that can remain static (E.g. the Output Compare Register1 A matching the value of Timer/Counter1) the interrupt flag is set when the event occurs. If the interrupt flag is cleared and the interrupt condition persists, the flag will not be set until the event occurs the next time.

MCU CONTROL REGISTER - MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - SRE : External SRAM Enable:
When the SRE bit is set (one), the external data SRAM is enabled, and the pin functions AD0-7 (Port A), A8-15 (Port C), $\overline{\text{WR}}$ and $\overline{\text{RD}}$ (Port D) are activated as the alternate pin functions. Then the SRE bit overrides any pin direction settings in the respective data direction registers. See "The SRAM Data Memory - Internal and External" for description of the External SRAM pin functions. When the SRE bit is cleared (zero), the external data SRAM is disabled, and the normal pin and data direction settings are used.

Bit 6 - SRW : External SRAM Wait State:
When the SRW bit is set (one), a one cycle wait state is inserted in the external data SRAM access cycle. When the SRW bit is cleared (zero), the external data SRAM access is executed with the normal two cycle scheme. See Figure 24 External Data SRAM Memory Cycles without Wait State and Figure 25: External Data SRAM Memory Cycles with Wait State.

Bit 5 - SE : Sleep Enable:
The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

Bit 4 - SM : Sleep Mode:
This bit selects between the two available sleep modes. When SM is cleared (zero), Idle Mode is selected as Sleep Mode. When SM is set (one), Power Down mode is selected as sleep mode. For details, refer to the paragraph "Sleep Modes" below.

Bit 3, 2 - ISC11, ISC10 : Interrupt Sense Control 1 bit 1 and bit 0:
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask in the GIMSK is set. The level and edges on the external INT1 pin that activate the interrupt are defined in the following table:

Table 4. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Note: When changing the ISC11/ISC10 bits, INT1 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bit 1, 0 - ISC01, ISC00 : Interrupt Sense Control 0 bit 1 and bit 0:
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask is set. The level and edges on the external INT0 pin that activate the interrupt are defined in the following table:

Table 5. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Note: When changing the ISC10/ISC00 bits, INT0 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Sleep Modes

To enter the sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM and I/O memory are unaltered. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset vector.

Note that if a *level* triggered interrupt is used for wake-up from power down, the low level must be held for a time longer than the oscillator start-up time of 16 ms. Otherwise, the interrupt flag may return to zero before the MCU starts executing.

IDLE MODE

When the SM bit is cleared (zero), the SLEEP instruction forces the MCU into the Idle Mode stopping the CPU but allowing Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like Timer Overflow interrupt and watchdog reset. If wakeup from the Analog Comparator interrupt is not required, the analog comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status register - ACSR. This will reduce power consumption during Idle Mode.

POWER DOWN MODE

When the SM bit is set (one), the SLEEP instruction forces the MCU into the Power Down Mode. In this mode, the external oscillator is stopped. The user can select whether the watchdog shall be enabled during power-down mode. If the watchdog is enabled, it will wake up the MCU when the Watchdog Time-out period expires. If the watchdog is disabled, only an external reset or an external level triggered interrupt can wake up the MCU.

Timer / Counters

The AT90S4414 provides two general purpose Timer/Counters - one 8-bit T/C and one 16-bit T/C. The Timer/Counters have individual prescaling selection from the same 10-bit prescaling timer. Both Timer/Counters can either be used as a timer with an internal clock timebase or as a counter with an external pin connection which triggers the counting.

The Timer/Counter Prescaler

Figure 32 shows the general Timer/Counter prescaler.

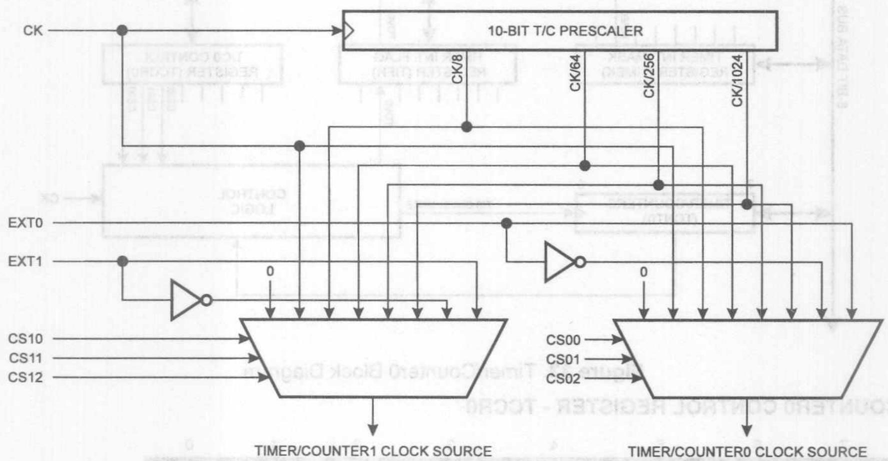


Figure 32. Timer/Counter Prescaler

The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. For the two Timer/Counters, added selections as CK, external source and stop, can be selected as clock sources.

The 8-Bit Timer/Counter0

Figure 33 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter0 Control Register - TCCR0. The overflow status flag is found in the Timer/Counter Insterrupt Flag Register - TIFR. Control signals are found in the Timer/Counter0 Control Register - TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

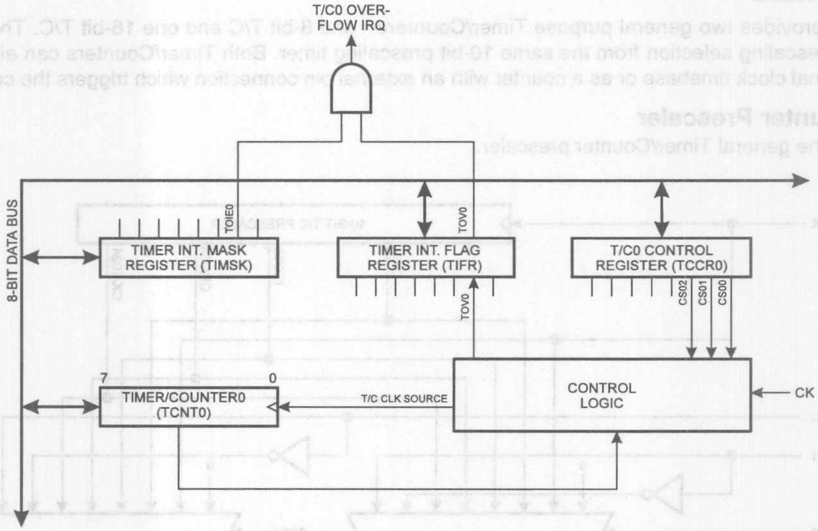


Figure 33. Timer/Counter0 Block Diagram

THE TIMER/COUNTER0 CONTROL REGISTER - TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	-	-	-	-	-	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7,6 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and always read zero.

Bits 2,1,0 - CS02, CS01, CS00 : Clock Select0, bit 2,1 and 0:
The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer0.

Table 6. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual data direction control register (cleared to zero gives an input pin).

THE TIMER COUNTER 0 - TCNT0

Bit	7	6	5	4	3	2	1	0	
$32 ($52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the clock cycle following the write operation.

The 16-Bit Timer/Counter1

Figure 34 shows the block diagram for Timer/Counter1.

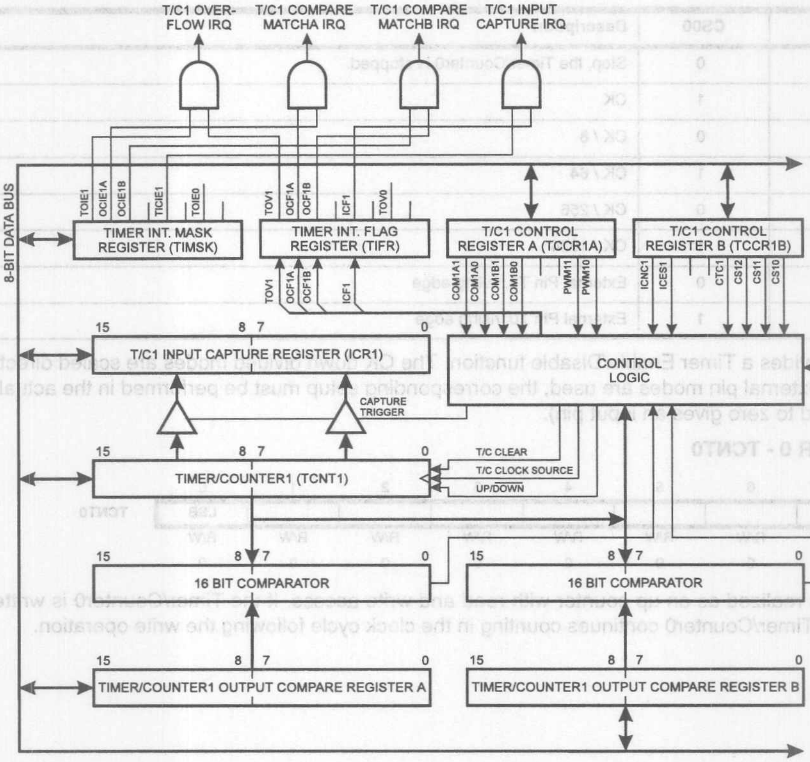


Figure 34. Timer/Counter1 Block Diagram

The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter1 Control Registers - TCCR1A and TCCR1B. The different status flags (overflow, compare match and capture event) and control signals are found in the Timer/Counter1 Control Registers - TCCR1A and TCCR1B. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 16-bit Timer/Counter1 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B - OCR1A and OCR1B as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as a 8, 9 or 10-bit Pulse With Modulator. In this mode the counter and the OCR1A/OCR1B registers serve as a dual glitch-free stand-alone PWM with centered pulses. Refer to Page 4-41 for a detailed description on this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register - ICR1, triggered by an external event on the Input Capture Pin - ICP. The actual capture event settings are defined by the Timer/Counter1 Control Register - TCCR1B. In addition, the Analog Comparator can be set to trigger the Input Capture. Refer to the section, “The Analog Comparator”, for details on this. The ICP pin logic is shown in Figure 35.

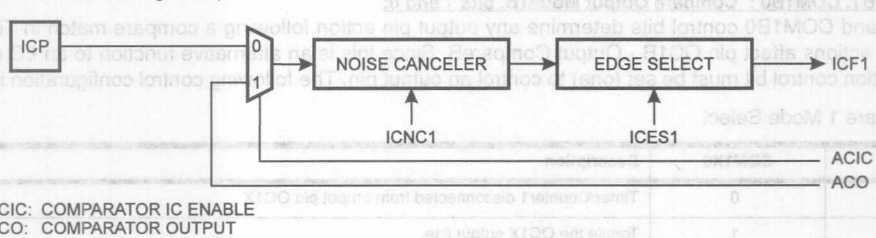


Figure 35. ICP Pin Schematic Diagram

The Timer/Counter1 input capture noise canceler block diagram is shown in Figure 36.

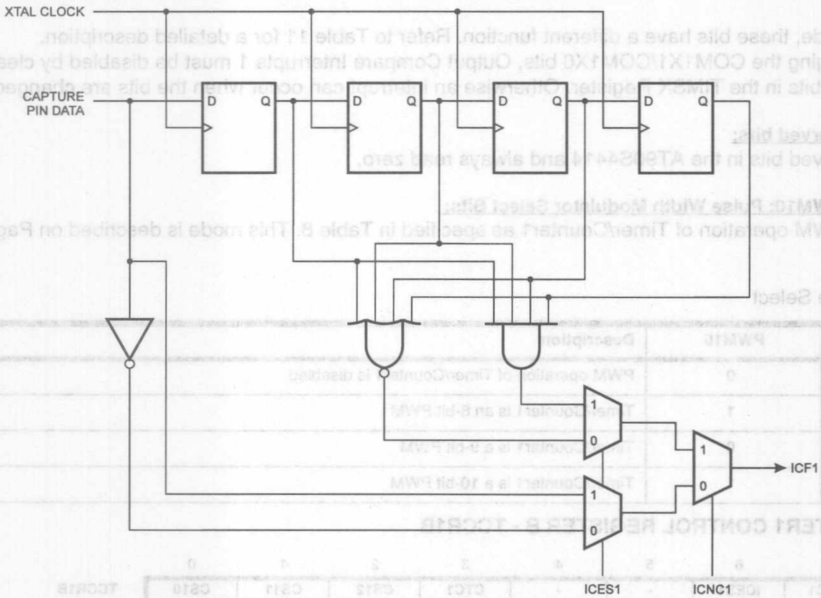


Figure 36. The Input Capture Noise Canceler

If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over 4 samples before the capture is activated. The input pin signal is sampled at XTAL clock frequency.

THE TIMER/COUNTER1 CONTROL REGISTER A - TCCR1A

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7,6 - COM1A1, COM1A0 : Compare Output Mode1A, bits 1 and 0:
The COM1A1 and COM1A0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1A - Output CompareA pin 1. Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 7.

Bits 5,4 - COM1B1, COM1B0 : Compare Output Mode1B, bits 1 and 0:
The COM1B1 and COM1B0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1B - Output CompareB. Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) to control an output pin. The following control configuration is given:

Table 7. Compare 1 Mode Select

COM1X1	COM1X0	Description
0	0	Timer/Counter1 disconnected from output pin OC1X
0	1	Toggle the OC1X output line.
1	0	Clear the OC1X output line (to zero).
1	1	Set the OC1X output line (to one).

X = A or B

In PWM mode, these bits have a different function. Refer to Table 11 for a detailed description.
When changing the COM1X1/COM1X0 bits, Output Compare Interrupts 1 must be disabled by clearing their Interrupt Enable bits in the TIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bits 3..2 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and always read zero.

Bits 1..0 - PWM11, PWM10: Pulse Width Modulator Select Bits:
These bits select PWM operation of Timer/Counter1 as specified in Table 8. This mode is described on Page 4-41.

Table 8. PWM Mode Select

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

THE TIMER/COUNTER1 CONTROL REGISTER B - TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/w	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ICNC1 : Input Capture1 Noise Canceler (4 CKs):
When the ICNC1 bit is cleared (zero), the input capture trigger noise canceler function is disabled. The input capture is triggered at the first rising/falling edge sampled on the ICP - input capture pin - as specified. When the ICNC1 bit is set (one), four successive samples are measures on the ICP - input capture pin, and all samples must be high/low according to the input capture trigger specification in the ICES1 bit. The actual sampling frequency is XTAL clock frequency.

Bit 6 - ICES1 : Input Capture1 Edge Select:
While the ICES1 bit is cleared (zero), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the falling edge of the input capture pin - ICP. While the ICES1 bit is set (one), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the rising edge of the input capture pin - ICP.

Bits 5, 4 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and always read zero.

Bit 3 - CTC1 : Clear Timer/Counter1 on Compare match:
When the CTC1 control bit is set (one), the Timer/Counter1 is reset to $0000 in the clock cycle after a compareA match. If the CTC1 control bit is cleared, the Timer/Counter1 continues counting until it is stopped, cleared, wraps around (overflow) or changes direction. In PWM mode, this bit has no effect.

Bits 2,1,0 - CS12, CS11, CS10 : Clock Select1, bit 2,1 and 0:
The Clock Select1 bits 2,1 and 0 define the prescaling source of Timer/Counter1.

Table 9. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual direction control register (cleared to zero gives an input pin).

THE TIMER/COUNTER1 - TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP).

- **TCNT1 Timer/Counter1 Write:**

  When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP register, and all 16 bits are written to the TCNT1 Timer/Counter1 register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.

- **TCNT1 Timer/Counter1 Read:**
  When the CPU reads the low byte TCNT1L, the data of the low byte TCNT1L is sent to the CPU and the data of the high byte TCNT1H is placed in the TEMP register. When the CPU reads the data in the high byte TCNT1H, the CPU receives the data in the TEMP register. Consequently, the low byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

TIMER/COUNTER1 OUTPUT COMPARE REGISTER - OCR1AH AND OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

TIMER/COUNTER1 OUTPUT COMPARE REGISTER - OCR1BH AND OCR1BL

Bit	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The output compare registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare Registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status register.

Since the Output Compare Registers - OCR1A and OCR1B - are 16-bit registers, a temporary register TEMP is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP register. When the CPU writes the low byte, OCR1AL or OCR1BL, the TEMP register is simultaneously written to OCR1AH or OCR1BH. Consequently, the high byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

THE TIMER/COUNTER1 INPUT CAPTURE REGISTER - ICR1H AND ICR1L

Bit	15	14	13	12	11	10	9	8	
$25 ($45)	MSB								ICR1H
$24 ($44)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The input capture register is a 16-bit read-only register.

When the rising or falling edge (according to the input capture edge setting - ICES1) of the signal at the input capture pin - ICP - is detected, the current value of the Timer/Counter1 is transferred to the Input Capture Register - ICR1. At the same time, the input capture flag - ICF1 - is set (one).

Since the Input Capture Register - ICR1 - is a 16-bit register, a temporary register TEMP is used when ICR1 is read to ensure that both bytes are read simultaneously. When the CPU reads the low byte ICR1L, the data is sent to the CPU and the data of the high byte ICR1H is placed in the TEMP register. When the CPU reads the data in the high byte ICR1H, the CPU receives the data in the TEMP register. Consequently, the low byte ICR1L must be accessed first for a full 16-bit register read operation.

TIMER/COUNTER1 IN PWM MODE

When the PWM mode is selected, Timer/Counter1 and the Output Compare Register1A - OCR1A and the Output Compare Register1B - OCR1B, form a dual 8, 9 or 10-bit, free-running, glitch-free and phase correct PWM with outputs on the PD5(OC1A) and OC1B pins. Timer/Counter1 acts as an up/down counter, counting up from \$0000 to TOP (see Table 10) , when it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the 10 least significant bits of OCR1A or OCR1B, the PD5(OC1A)/OC1B pins are set or cleared according to the settings of the COM1A1/COM1A0 or COM1B1/COM1B0 bits in the Timer/Counter1 Control Register TCCR1A. Refer to Table 11 for details.

Table 10. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP value	Frequency
8-bit	\$00FF (255)	$f_{TC1}/510$
9-bit	\$01FF (511)	$f_{TC1}/1022$
10-bit	\$03FF(1023)	$f_{TC1}/2046$

Table 11. Compare1 Mode Select in PWM Mode

COM1X1	COM1X0	Effect on OCX1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, upcounting. Set on compare match, downcounting (non-inverted PWM).
1	1	Cleared on compare match, downcounting. Set on compare match, upcounting (inverted PWM).

Note: X = A or B

Note that in the PWM mode, the 10 least significant OCR1A/OCR1B bits, when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 37 for an example.

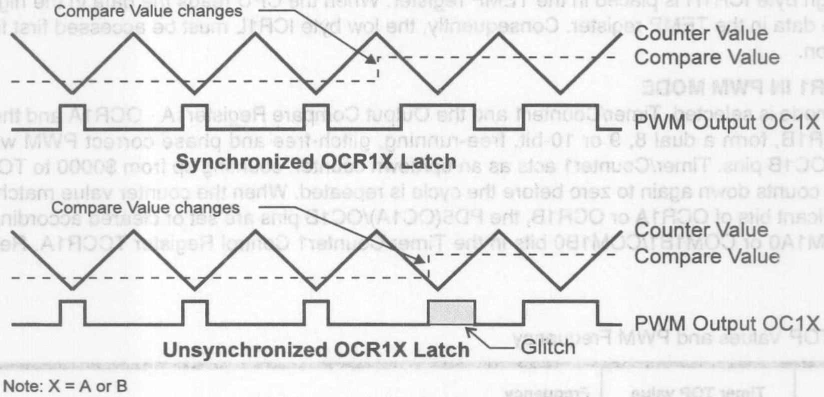


Figure 37. Effects on Unsynchronized OCR1 Latching

When OCR1 contains $0000 or TOP, the output OC1A/OC1B is held low or high according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 12:

Table 12. PWM Outputs OCR1X = $0000 or TOP

COM1X1	COM1X0	OCR1X	Output OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

Note: X = A or B

In PWM mode, the Timer Overflow Flag1, TOV1, is set when the counter changes direction at $0000. Timer Overflow Interrupt1 operates exactly as in normal Timer/Counter mode, i.e. it is executed when TOV1 is set provided that Timer Overflow Interrupt1 and global interrupts are enabled. This does also apply to the Timer Output Compare1 flags and interrupts.

The Watchdog Timer

The Watchdog Timer is clocked from a separate on-chip oscillator which runs at 1MHz This is the typical value at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted from 16 to 2048 ms. The WDR - Watchdog Reset - instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog reset, the AT90S4414 resets and executes from the reset vector. For timing details on the Watchdog reset, refer to Page 4-28.

To prevent unintentional disabling of the watchdog, a special turn-off secuence must be followed when the watchdog is disabled.Refer to the description of the Watchdog Timer Control Regiter for details

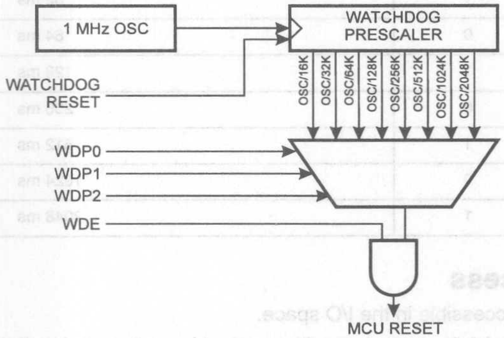


Figure 38. Watchdog Timer

THE WATCHDOG TIMER CONTROL REGISTER - WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	-	-	-	WDTTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..5 - Res : Reserved bits:

These bits are reserved bits in the AT90S4414 and will always read as zero.

Bit 4 - WDTOE : Watch Dog Turn-Off Enable:

This bit must be set (one) when the WDE bit is cleared. Otherwise, the watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure.

Bit 3 - WDE : Watch Dog Enable:

When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logcal one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logical 0 to WDE. This disables the watchdog.

Bits 2..0 - WDP2, WDP1, WDP0 : Watch Dog Timer Prescaler 2, 1 and 0:
The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 13.

Table 13. Watch Dog Timer Prescale Select (Typical Values at V_{CC} = 5V)

WDP2	WDP1	WDP0	Timeout Period
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written. An EEPROM brown-out detection prevents writing to the EEPROM if V_{CC} is below a certain level.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read or written, the CPU is halted for two clock cycles before the next instruction is executed.

THE EEPROM ADDRESS REGISTER - EEAR

Bit	7	6	5	4	3	2	1	0	
$1E ($3E)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The EEPROM Address Registers - EEARH and EEARL specify the EEPROM address in the 256 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 255.

THE EEPROM DATA REGISTER - EEDR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..0 - EEDR7..0 : EEPROM Data:
For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

THE EEPROM CONTROL REGISTER - EECR

Bit	7	6	5	4	3	2	1	0	
\$1C (\$3C)	-	-	-	-	-	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7..3 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and will always read as zero.

Bit 2 - EEMWE : EEPROM Master Write Enable:
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for a EEPROM write procedure.

Bit 1 - EEWE : EEPROM Write Enable:
The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is unessential):

1. Wait until EEWE becomes zero.
2. Write new EEPROM address to EEAR (optional)
3. Write new EEPROM data to EEDR (optional)
4. Write a logical one to the EEMWE bit in EECR
5. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

When the write access time (typically 2.5 ms at V_{CC} = 5V or 4 ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 - EERE : EEPROM Read Enable:
The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress when new data or address is written to the EEPROM I/O registers, the write operation will be interrupted, and the result is undefined.

The Serial Peripheral Interface - SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT90S4414 and peripheral devices or between several AT90S4414 devices. The AT90S4414 SPI features include the following:

- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 5 Mbit/s Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

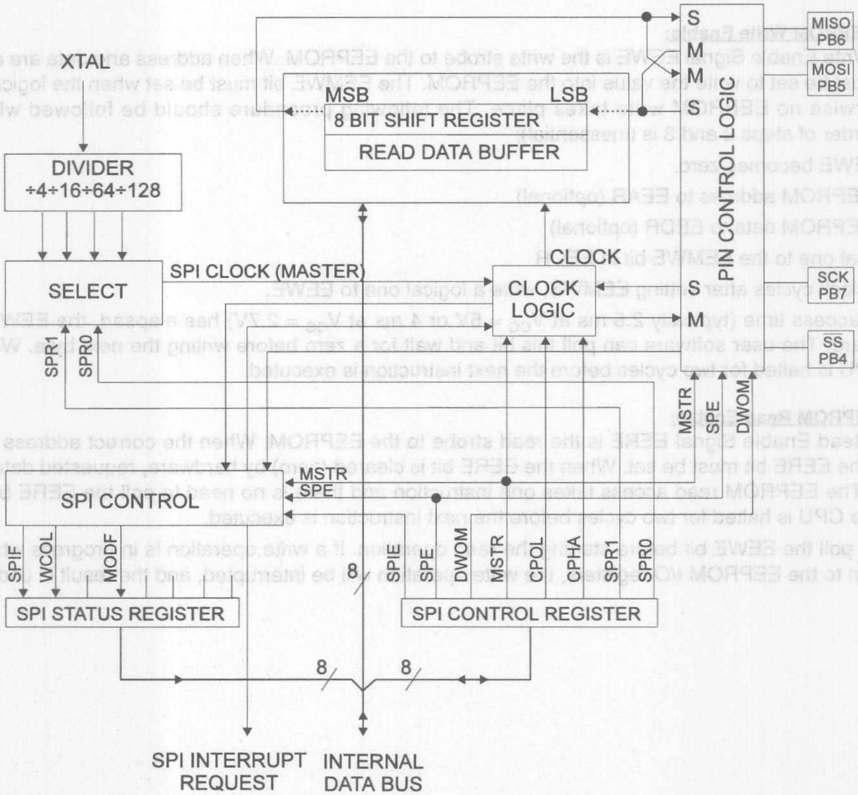


Figure 39. SPI Block Diagram

The interconnection between master and slave CPUs with SPI is shown in Figure 40. The PB7(SCK) pin is the clock output in the master mode and is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR register becomesset, an interrupt is requested. The Slave Select input, PB4($\overline{SS}$), is set low to select an individual SPI device as a slave. The two shift registers in the Master and the Slave can be considered as one distributed 16-bit circular shift register. This is shown in Figure 40. When data is shifted from the master to the slave, data is also shifted in the opposite direction, simultaneously. This means that during one shift cycle, data in the master and the slave are interchanged.

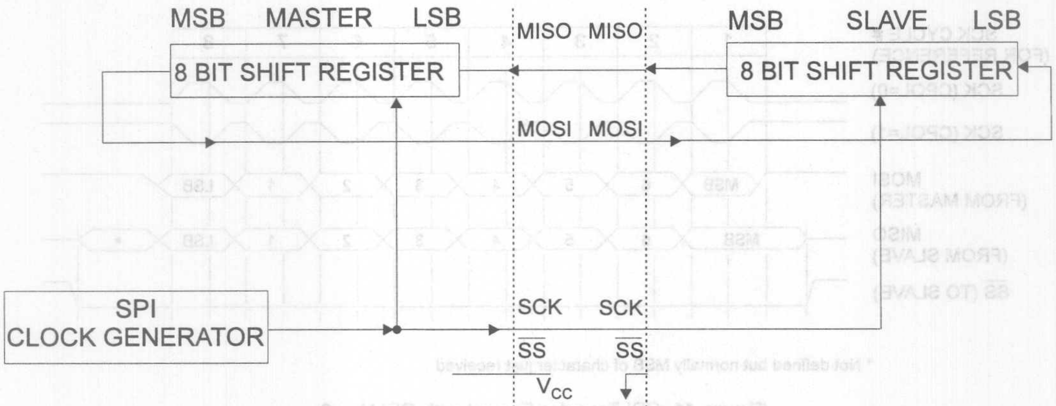


Figure 40. SPI Master-Slave Interconnection

The system is single buffered in the transmit direction and double buffered in the receive direction. This means that characters to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first character is lost.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK and $\overline{SS}$ pins is overriden according to the following table:

Table 14. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

$\overline{SS}$ Pin Functionality

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin. If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. If $\overline{SS}$ is configured as an input, it must be hold high to ensure Master SPI operatin. If, in master mode, the $\overline{SS}$ pin is input, and is driven low by peripheral circuitry, the SPI system interpretes this as that another master selects the SPI as a slave and will start sending data to it. To avoid bus contention, the SPI system takes the following actions:

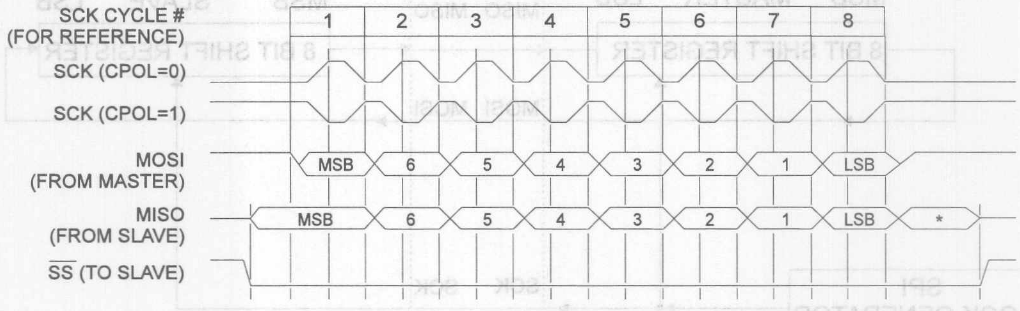
1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmittal is used in master mode, and there exists a possibility that SS is driven low, the interrupt should always check that the MSTR bit is still set. Once the MSTR bit has been cleared by a slave select, it must be set by the user.

When the SPI is configured as a slave, the $\overline{SS}$ is always input. When $\overline{SS}$ is held low, the SPI is activated and MISO becomes an output if configured so by the user. All other pins are inputs When $\overline{SS}$ is driven low, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data.

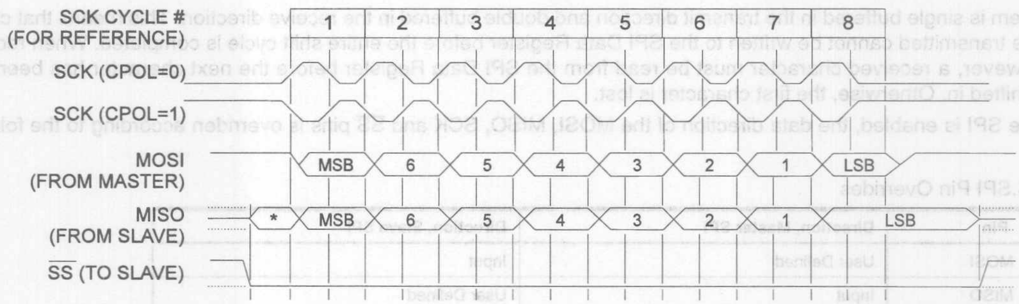
Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 41 and Figure 42.



* Not defined but normally MSB of character just received

Figure 41. SPI Transfer Format with CPHA = 0



* Not defined but normally LSB of previously transmitted character

Figure 42. SPI Transfer Format with CPHA = 1

THE SPI CONTROL REGISTER - SPCR

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	1	0	0	

Bit 7 - SPIE : SPI Interrupt Enable:
This bit causes setting of the SPIF bit in the SPSR register to execute the SPI interrupt provided that global interrupts are enabled.

Bit 6 - SPE : SPI Enable:
When the SPE bit is set (one), the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 - DORD : Data ORDer:
When the DORD bit is set (one), the LSB of the data word is transmitted first.

When the DORD bit is cleared (zero), the MSB of the data word is transmitted first.

Bit 4 - MSTR : Master/Slave Select:
This bit selects Master SPI mode when set (one), and Slave SPI mode when cleared (zero). If $\overline{SS}$ is configured as an input and is driven low whil MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI master mode.

4

Bit 3 - CPOL : Clock POLarity:
When this bit is set (one), SCK is high when idle. When CPOL is cleared (zero), SCK is low when idle. Refer to Figure 41 and Figure 42 for additional information.

Bit 2 - CPHA : Clock PHAse:
Refer to Figure 41 or Figure 42 for the functionality of this bit.

Bits 1,0 - SPR1, SPR0 : SPI Clock Rate Select 1 and 0:
These two bits control the SCK rate of the device configured as a master. SPR1 and SPR2 have no effect on the slave. The relationship between SCK and the Oscillator Clock frequency f_{cl} is shown in the following table:

Table 15. Relationship Between SCK and the Oscillator Frequency

SPR1	SPR0	SCK Frequency
0	0	f_{cl} / 4
0	1	f_{cl} / 16
1	0	f_{cl} / 64
1	1	f_{cl} / 128

THE SPI STATUS REGISTER - SPSR

Bit	7	6	5	4	3	2	1	0	
$0E ($2E)	SPIF	WCOL	-	-	-	-	-	-	SPSR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - SPIF : SPI Interrupt Flag:
When a serial transfer is complete, the SPIF bit is set (one) and an interrupt is generated if SPIE in SPCR is set (one) and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in master mode, this will also set the SPIF

flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI status register with SPIF set (one), then accessing the SPI Data Register (SPDR).

Bit 6 - WCOL : Write COLlision flag:
The WCOL bit is set if the SPI data register (SPDR) is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it will have no effect. The WCOL bit (and the SPIF bit) are cleared (zero) by first reading the SPI Status Register with WCOL set (one), and then accessing the SPI Data Register.

Bit 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and will always read as zero.

The SPI interface on the AT90S4414 is also used for program memory and EEPROM downloading or uploading. See Page 4-78 for serial programming and verification.

THE SPI DATA REGISTER - SPDR

Bit	7	6	5	4	3	2	1	0	
$0F ($2F)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The SPI Data Register is a read/write register used for data transfer between the register file and the SPI Shift register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

The UART

The AT90S4414 features a full duplex Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- Baud rate generator generates any baud rate
- High baud rates at low XTAL frequencies
- 8 or 9 bits data
- Noise filtering
- Overrun detection
- Framing Error detection
- False Start Bit detection
- Three separate interrupts on TX Complete, TX Data Register Empty and RX Complete

Data Transmission

A block schematic of the UART transmitter is shown in Figure 43.

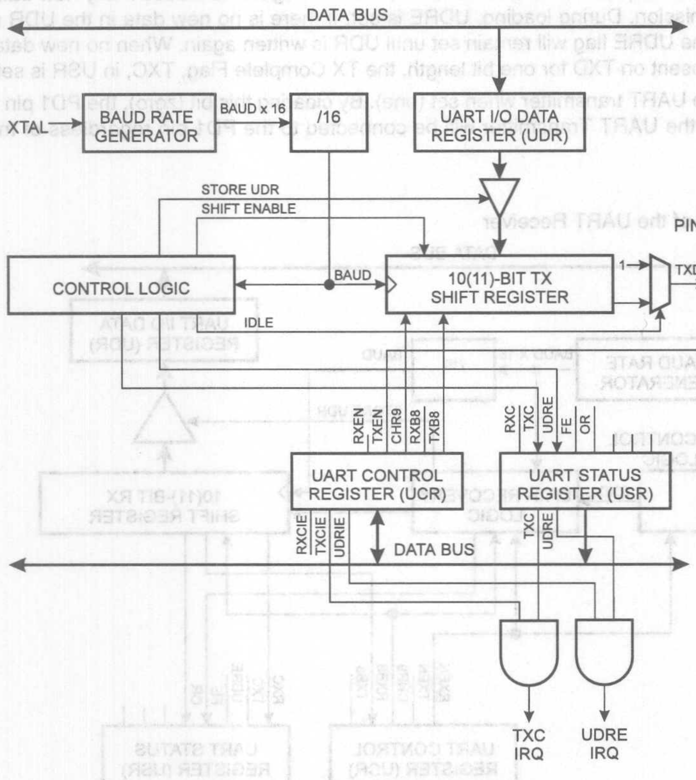


Figure 43. UART Transmitter

Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDR. Data is transferred from UDR to the Transmit shift register when:

- A new character has been written to UDR after the stop bit from the previous character has been shifted out. The shift register is loaded immediately.
- A new character has been written to UDR before the stop bit from the previous character has been shifted out. The shift register is loaded when the stop bit of the character currently being transmitted has been shifted out.

If the 10(11)-bit Transmitter shift register is empty or when, data is transferred from UDR to the shift register. At this time the UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the data is transferred from UDR to the 10(11)-bit shift register, bit 0 of the shift register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit shift register.

4

On the Baud Rate clock following the transfer operation to the shift register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the shift register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR register to send when the stop bit is shifted out, the UDRE flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the TX Complete Flag, TXC, in USR is set.

The TXEN bit in UCR enables the UART transmitter when set (one). By clearing this bit (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to the PD1 pin regardless of the setting of the DDD1 bit in DDRB.

Data Reception

Figure 44 shows a block diagram of the UART Receiver

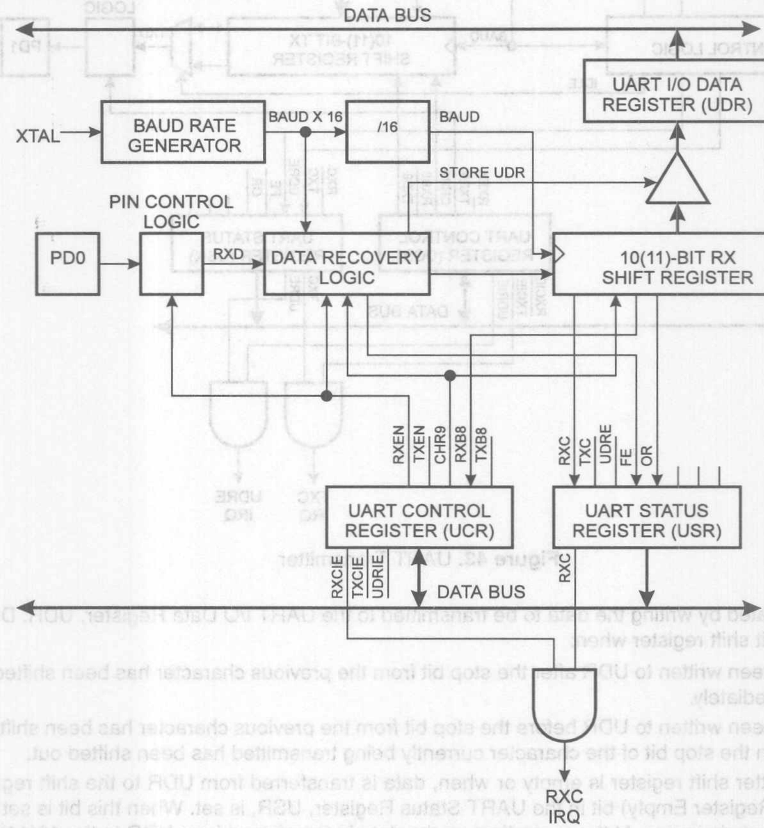


Figure 44. UART Receiver

The receiver front-end logic samples the signal on the RXD pin at a frequency 16 times the baud rate. While the line is idle, one single sample of logical zero will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1 to 0-transition, the receiver samples the RXD pin at samples 8, 9 and 10. If two or more of these three samples are found to be logical ones, the start bit is rejected as a noise spike and the receiver starts looking for the next 1 to 0-transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 8, 9 and 10. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the transmitter shift register as they are sampled. Sampling of an incoming character is shown in Figure 45.

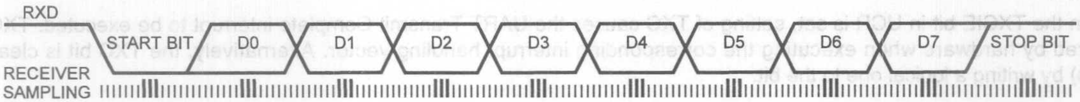


Figure 45. Sampling Received Data

When the stop bit enters the receiver, the majority of the three samples must be one to accept the stop bit. If two or more samples are logical zeros, the Framing Error (FE) flag in the UART Status Register (USR) is set. Before reading the UDR register, the user should always check the FE bit to detect Framing Errors.

Whether or not a valid stop bit is detected at the end of a character reception cycle, the data is transferred to UDR and the RXC flag in USR is set. UDR is in fact two physically separate registers, one for transmitted data and one for received data. When UDR is read, the Receive Data register is accessed, and when UDR is written, the Transmit Data register is accessed. If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the RXB8 bit in UCR is loaded with bit 9 in the Transmit shift register when data is transferred to UDR.

If, after having received a character, the UDR register has not been read since the last receive, the OverRun (OR) flag in UCR is set. This means that the last data byte shifted into to the shift register could not be transferred to UDR and has been lost. The OR bit is buffered, and is updated when the valid data byte in UDR is read. Thus, the user should always check the OR bit after reading the UDR register in order to detect any overruns.

By clearing the RXEN bit in the UCR register, the receiver is disabled. This means that the PD0 pin can be used as a general I/O pin. When RXEN is set, the UART Receiver will be connected to the PD0 pin regardless of the setting of the DDD0 bit in DDRB.

UART Control

THE UART I/O DATA REGISTER - UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UDR register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDR, the UART Receive Data register is read.

THE UART STATUS REGISTER - USR

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	OR	-	-	-	USR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	1	0	0	0	0	0	

The USR register is a read-only register providing information on the UART Status.

Bit 7 - RXC: UART Receive Complete:
This bit is set (one) when a received character is transferred from the Receiver Shift register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set(one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

Bit 6 - TXC : UART Transmit Complete:
This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical one to the bit.

Bit 5 - UDRE : UART Data Register Empty:
This bit is set (one) when a character written to UDR is transferred to the Transmit shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCR is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.

Bit 4 - FE : Framing Error:
This bit is set if a Framing Error condition is detected, i.e. when the stop bit of an incoming character is zero.

The FE bit is cleared when the stop bit of received data is one.

Bit 3 - OR : OverRun:
This bit is set if an Overrun condition is detected, i.e. when a character already present in the UDR register is not read before the next character has been shifted into the Receiver Shift register. The OR bit is buffered, which means that it will be set once the valid data still in UDRE is read.

The OR bit is cleared (zero) when data is received and transferred to UDR.

Bits 2..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S4414 and will always read as zero.

THE UART CONTROL REGISTER - UCR

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - RXCIE : RX Complete Interrupt Enable:
When this bit is set (one), a setting of the RXC bit in USR will cause the Receive Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 6 - TXCIE : TX Complete Interrupt Enable:
When this bit is set (one), a setting of the TXC bit in USR will cause the Transmit Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 5 - UDRIE : UART Data Register Empty Interrupt Enable:
When this bit is set (one), a setting of the UDRE bit in USR will cause the UART Data Register Empty interrupt routine to be executed provided that global interrupts are enabled.

Bit 4 - RXEN : Receiver Enable:
This bit enables the UART receiver when set (one). When the receiver is disabled, the TXC, OR and FE status flags cannot become set. If these flags are set, turning off RXEN does not cause them to be cleared.

Bit 3 - TXEN : Transmitter Enable:
This bit enables the UART transmitter when set (one). When disabling the transmitter while transmitting a character, the transmitter is not disabled before the character in the shift register plus any following character in UDR has been completely transmitted.

Bit 2 - CHR9 : 9 Bit Characters:
When this bit is set (one) transmitted and received characters are 9 bit long plus start and stop bits. The 9th bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The 9th data bit can be used as an extra stop bit or a parity bit.

Bit 1 - RXB8 : Receive Data Bit 8
When CHR9 is set (one), RXB8 is the 9th data bit of the received character.

Bit 0 - TXB8 : Transmit Data Bit 8
When CHR9 is set (one), TXB8 is the 9th data bit in the character to be transmitted.

THE BAUD RATE GENERATOR

The baud rate generator is a frequency divider which generates baud-rates according to the following equation:

$$\mathrm{BAUD} = \frac{fck}{16(\mathrm{UBRR} + 1)}$$

- BAUD = Baud-Rate
- fck= Crystal Clock frequency
- UBRR= Contents of the UART Baud Rate register, UBRR (0-255)

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBRR settings in Table 16. UBRR values which yield an actual baud rate differing less than 2% from the target baud rate, are bolded in the table.

4

Table 16. UBRR Settings at Various Crystal Frequencies

Baud Rate	1 MHz	%Error	1.8432 MHz	%Error	2 MHz	%Error	2.4576 MHz	%Error
2400	UBRR= 25	0.2	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 63	0.0
4800	UBRR= 12	0.2	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 31	0.0
9600	UBRR= 6	7.5	UBRR= 11	0.0	UBRR= 12	0.2	UBRR= 15	0.0
14400	UBRR= 3	7.8	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 10	3.1
19200	UBRR= 2	7.8	UBRR= 5	0.0	UBRR= 6	7.5	UBRR= 7	0.0
28800	UBRR= 1	7.8	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	6.3
57600	UBRR= 0	7.8	UBRR= 1	0.0	UBRR= 1	7.8	UBRR= 2	12.5
115200	UBRR= 0	84.3	UBRR= 0	0.0	UBRR= 0	7.8	UBRR= 0	25.0

Baud Rate	3.2768 MHz	%Error	3.6864 MHz	%Error	4 MHz	%Error	4.608 MHz	%Error
2400	UBRR= 84	0.4	UBRR= 95	0.0	UBRR= 103	0.2	UBRR= 119	0.0
4800	UBRR= 42	0.8	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 59	0.0
9600	UBRR= 20	1.6	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 29	0.0
14400	UBRR= 13	1.6	UBRR= 15	0.0	UBRR= 16	2.1	UBRR= 19	0.0
19200	UBRR= 10	3.1	UBRR= 11	0.0	UBRR= 12	0.2	UBRR= 14	0.0
28800	UBRR= 6	1.6	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 9	0.0
57600	UBRR= 3	12.5	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	0.0
115200	UBRR= 1	12.5	UBRR= 1	0.0	UBRR= 1	7.8	UBRR= 2	20.0

Baud Rate	7.3728 MHz	%Error	8 MHz	%Error	9.216 MHz	%Error	11.059 MHz	%Error
2400	UBRR= 191	0.0	UBRR= 207	0.2	UBRR= 239	0.0	UBRR= 287	-
4800	UBRR= 95	0.0	UBRR= 103	0.2	UBRR= 119	0.0	UBRR= 143	0.0
9600	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 59	0.0	UBRR= 71	0.0
14400	UBRR= 31	0.0	UBRR= 34	0.8	UBRR= 39	0.0	UBRR= 47	0.0
19200	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 29	0.0	UBRR= 35	0.0
28800	UBRR= 15	0.0	UBRR= 16	2.1	UBRR= 19	0.0	UBRR= 23	0.0
57600	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 9	0.0	UBRR= 11	0.0
115200	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	0.0	UBRR= 5	0.0

Baud Rate	14.746 MHz	%Error	16 MHz	%Error	18.432 MHz	%Error	20 MHz	%Error
2400	UBRR= 383	-	UBRR= 416	-	UBRR= 479	-	UBRR= 520	-
4800	UBRR= 191	0.0	UBRR= 207	0.2	UBRR= 239	0.0	UBRR= 259	-
9600	UBRR= 95	0.0	UBRR= 103	0.2	UBRR= 119	0.0	UBRR= 129	0.2
14400	UBRR= 63	0.0	UBRR= 68	0.6	UBRR= 79	0.0	UBRR= 86	0.2
19200	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 59	0.0	UBRR= 64	0.2
28800	UBRR= 31	0.0	UBRR= 34	0.8	UBRR= 39	0.0	UBRR= 42	0.9
57600	UBRR= 15	0.0	UBRR= 16	2.1	UBRR= 19	0.0	UBRR= 21	1.4

THE UART BAUD RATE REGISTER - UBRR

Bit	7	6	5	4	3	2	1	0	
$09 ($29)	MSB							LSB	UBRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UBRR register is an 8-bit read/write register which specifies the UART Baud Rate according to the equation on the previous page.

The Analog Comparator

The analog comparator compares the input values on the positive pin PB2 (AIN0) and negative pin PB3 (AIN1). When the voltage on the positive pin PB2 (AIN0) is higher than the voltage on the negative pin PB3 (AIN1), the Analog Comparator Output, ACO is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 46.

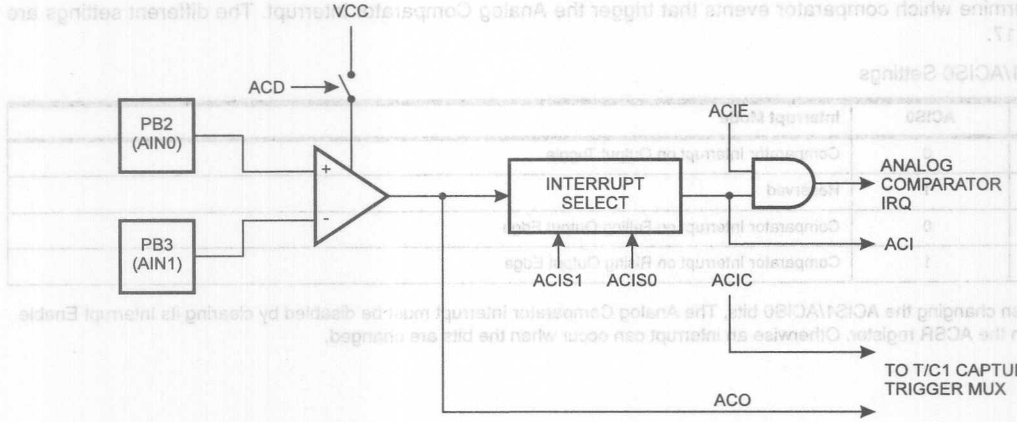


Figure 46. Analog Comparator Block Diagram

THE ANALOG COMPARATOR CONTROL AND STATUS REGISTER - ACSR

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ACD : Analog Comparator Disable
When this bit is set(one), the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. It is most commonly used if power consumption during Idle Mode is critical, and wake-up from the analog comparator is not required. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 - Res : Reserved bit:
This bit is a reserved bit in the AT90S4414 and will always read as zero.

Bit 5 - ACO : Analog Comparator Output:
ACO is directly connected to the comparator output.

Bit 4 - ACI : Analog Comparator Interrupt Flag:
This bit is set (one) when a comparator output event triggers the interrupt mode defined by ACI1 and ACI0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set (one) and the I-bit in SREG is set (one). ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 - ACIE : Analog Comparator Interrupt Enable:
When the ACIE bit is set (one) and the I-bit in the Status Register is set (one), the analog comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

Bit 2 - ACIC : Analog Comparator Input Capture enable:
When set (one), this bit enables the Input Capture function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When cleared (zero), no connection between the analog comparator and the Input Capture function is given. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set (one).

Bits 1,0 - ACIS1, ACIS0 : Analog Comparator Interrupt Mode Select:
These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 17.

Table 17. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

Note: When changing the ACIS1/ACIS0 bits, The Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

I/O-Ports

Port A

PORT A is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port A, one each for the Data Register - PORTA, $1B($3B), Data Direction Register - DDRA, $1A($3A) and the Port A Input Pins - PINA, $19($39). The Port A Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The PORT A output buffers can sink 20mA and thus drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The PORT A pins have alternate functions related to the optional external data SRAM. PORT A can be configured to be the multiplexed low-order address/data bus during accesses to the external data memory. In this mode, PORT A has internal pullups.

When PORT A is set to the alternate function by the SRE - External SRAM Enable - bit in the MCUCR - MCU Control Register, the alternate settings override the data direction register.

THE PORT A DATA REGISTER - PORTA

Bit	7	6	5	4	3	2	1	0	
\$1B (\$3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT A DATA DIRECTION REGISTER - DDRA

Bit	7	6	5	4	3	2	1	0	
\$1A (\$3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT A INPUT PINS ADDRESS - PINA

Bit	7	6	5	4	3	2	1	0	
\$19 (\$39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port A Input Pins address - PINA - is not a register, and this address enables access to the physical value on each Port A pin. When reading PORTA the PORTA Data Latch is read, and when reading PINA, the logical values present on the pins are read.

PORTA AS GENERAL DIGITAL I/O

All 8 bits in PORT A are equal when used as digital I/O pins.

PAn, General I/O pin: The DDAn bit in the DDRA register selects the direction of this pin, if DDAn is set (one), PAn is configured as an output pin. If DDAn is cleared (zero), PAn is configured as an input pin. If PORTAn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTAn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 18. DDAn Effects on PORT A Pins

DDAn	PORTAn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PAn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

PORT A SCHEMATICS

Note that all port pins are synchronized. The synchronization latch is however, not shown in the figure.

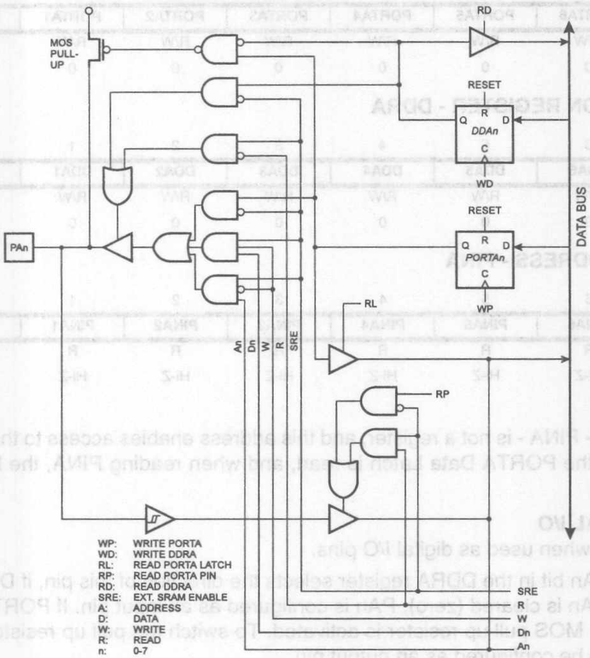


Figure 47. PORTA Schematic Diagrams (Pins PA0 - PA7)

Port B

Port B is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port B, one each for the Data Register - PORTB, $18($38), Data Direction Register - DDRB, $17($37) and the Port B Input Pins - PINB, $16($36). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The Port B output buffers can sink 20mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The Port B pins with alternate functions are shown in the following table:

Table 19. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	T0 (Timer/Counter 0 external counter input)
PB1	T1 (Timer/Counter 1 external counter input)
PB2	AIN0 (Analog comparator positive input)
PB3	AIN1 (Analog comparator negative input)
PB4	$\overline{SS}$ (SPI Slave Select input)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

When the pins are used for the alternate function the DDRB and PORTB register has to be set according to the alternate function description.

THE PORT B DATA REGISTER - PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B DATA DIRECTION REGISTER - DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B INPUT PINS ADDRESS - PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port B Input Pins address - PINB - is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the PORTB Data Latch is read, and when reading PINB, the logical values present on the pins are read.

PORTB AS GENERAL DIGITAL I/O

All 8 bits in port B are equal when used as digital I/O pins.

PBn, General I/O pin: The DDBn bit in the DDRB register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 20. DDBn Effects on Port B Pins

DDBn	PORTBn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PBn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

ALTERNATE FUNCTIONS OF PORTB

The alternate pin configuration is as follows:

SCK - PORTB, Bit 7:

SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit. See the description of the SPI port for further detatils.

MISO - PORTB, Bit 6:

MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit. See the description of the SPI port for further detatils.

MOSI - PORTB, Bit 5:

MOSI: SPI Master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further detatils.

SS - PORTB, Bit 4:

$\overline{SS}$: Slave port select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further detatils.

AIN1 - PORTB, Bit 3

AIN1, Analog Comparator Negative Input. When configured as an input (DDB3 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB3 is cleared (zero)), this pin also serves as the negative input of the on-chip analog comparator.

AIN0 - PORTB, Bit 2

AIN0, Analog Comparator Positive Input. When configured as an input (DDB2 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB2 is cleared (zero)), this pin also serves as the positive input of the on-chip analog comparator.

T1 - PORTB, Bit 1:

T1, Timer/Counter1 counter source. See the timer description for further details

T0 - PORTB, Bit 0:

T0: Timer/Counter0 counter source. See the timer description for further details.

PORT B SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

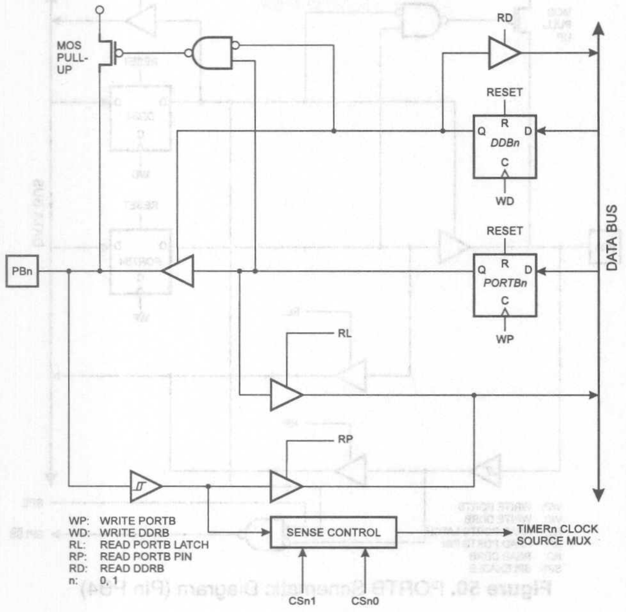


Figure 48. PORTB Schematic Diagram (Pins PB0 and PB1)

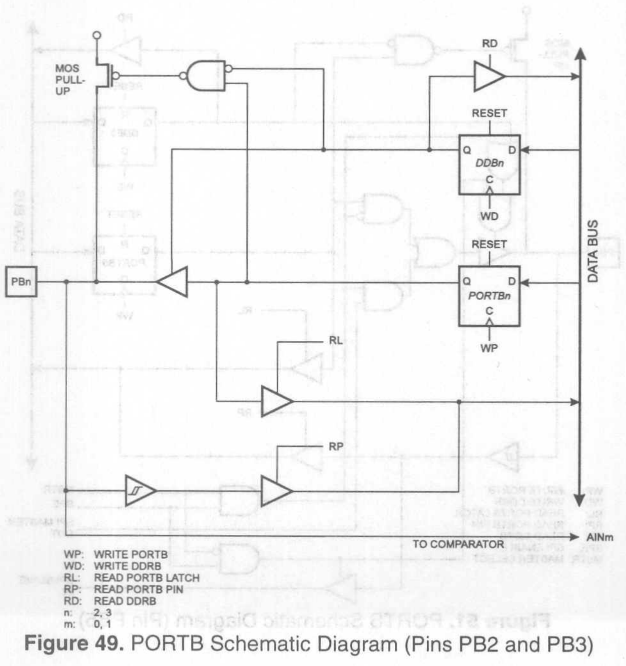


Figure 49. PORTB Schematic Diagram (Pins PB2 and PB3)

4

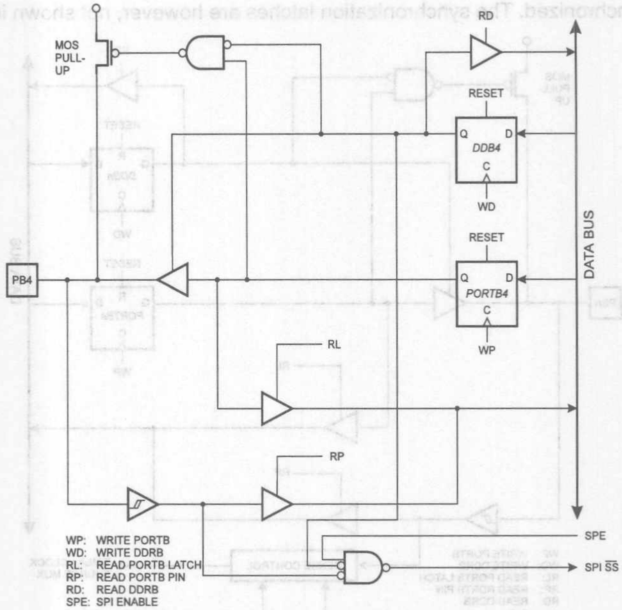


Figure 50. PORTB Schematic Diagram (Pin PB4)

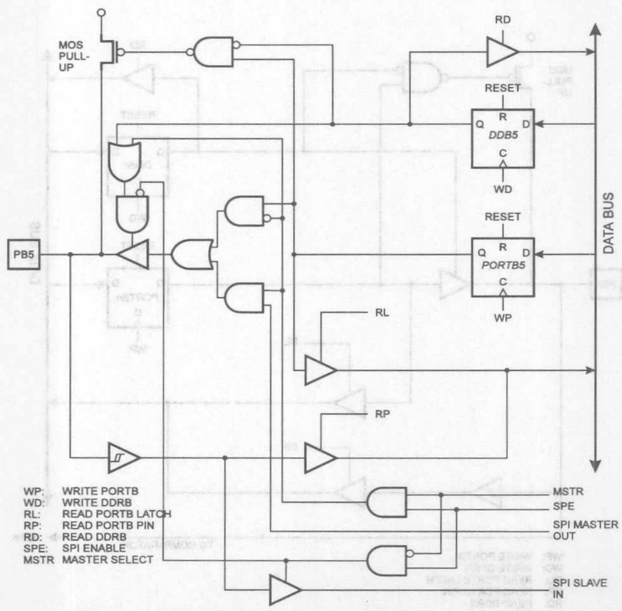


Figure 51. PORTB Schematic Diagram (Pin PB5)

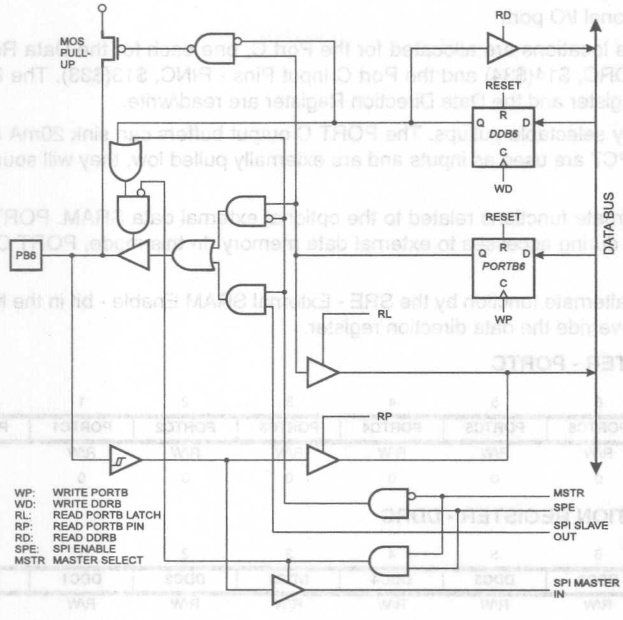


Figure 52. PORTB Schematic Diagram (Pin PB6)

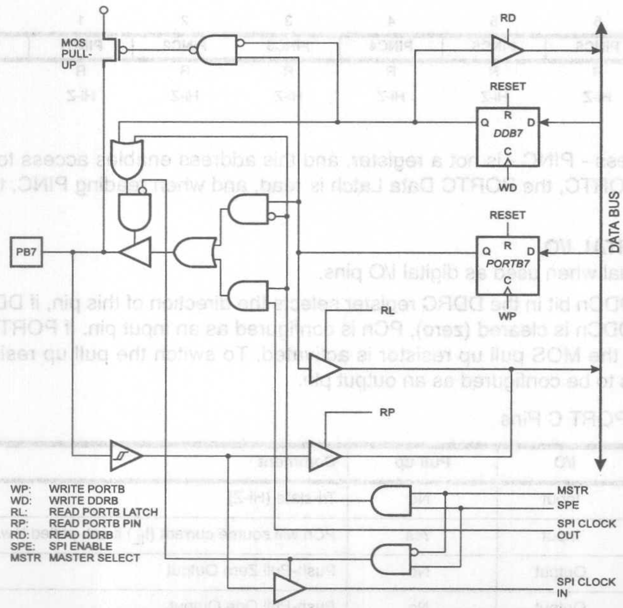


Figure 53. PORTB Schematic Diagram (Pin PB7)

Port C

PORT C is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port C, one each for the Data Register - PORTC, $15($35), Data Direction Register - DDRC, $14($34) and the Port C Input Pins - PINC, $13($33). The Port C Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The PORT C output buffers can sink 20mA and thus drive LED displays directly. When pins PC0 to PC7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The PORT C pins have alternate functions related to the optional external data SRAM. PORT C can be configured to be the high-order address byte during accesses to external data memory. In this mode, PORT C uses internal pullups when emitting 1's.

When PORT C is set to the alternate function by the SRE - External SRAM Enable - bit in the MCUCR - MCU Control Register, the alternate settings override the data direction register.

THE PORT C DATA REGISTER - PORTC

Bit	7	6	5	4	3	2	1	0	
$15 ($35)	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT C DATA DIRECTION REGISTER - DDRC

Bit	7	6	5	4	3	2	1	0	
$14 ($34)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT C INPUT PINS ADDRESS - PINC

Bit	7	6	5	4	3	2	1	0	
$13 ($33)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port C Input Pins address - PINC - is not a register, and this address enables access to the physical value on each Port C pin. When reading PORTC, the PORTC Data Latch is read, and when reading PINC, the logical values present on the pins are read.

PORTC AS GENERAL DIGITAL I/O

All 8 bits in PORT C are equal when used as digital I/O pins.

PCn, General I/O pin: The DDCn bit in the DDRC register selects the direction of this pin, if DDCn is set (one), PCn is configured as an output pin. If DDCn is cleared (zero), PCn is configured as an input pin. If PORTCn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, PORTCn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 21. DDCn Effects on PORT C Pins

DDCn	PORTCn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PCn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7…0, pin number

PORT C SCHEMATICS

Note that all port pins are synchronized. The synchronization latch is however, not shown in the figure.

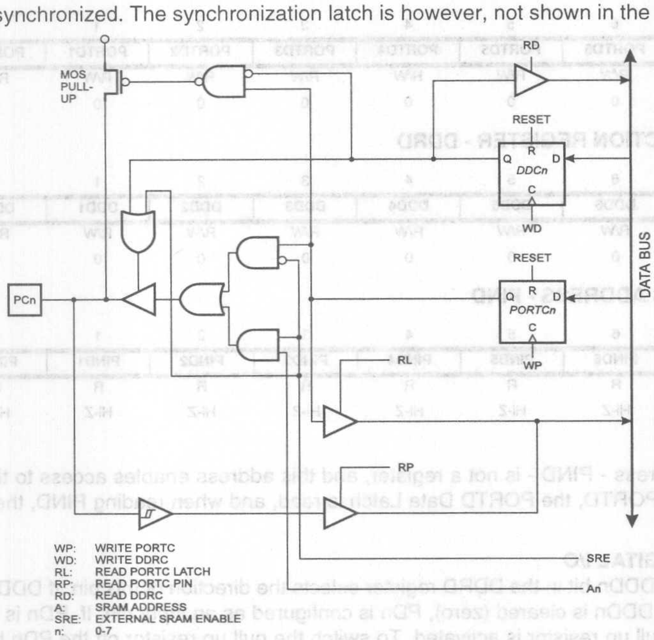


Figure 54. PORTC Schematic Diagram (Pins PC0 - PC7)

Port D

Port D is an 8 bit bi-directional I/O port with internal pullups.

Three data memory address locations are allocated for the Port D, one each for the Data Register - PORTD, $12($32), Data Direction Register - DDRD, $11($31) and the Port D Input Pins - PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Some Port D pins have alternate functions as shown in the following table:

Table 22. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RDX (UART Input line)
PD1	TDX (UART Output line)
PD2	INT0 (External interrupt 0 input)
PD3	INT1 (External interrupt 1 input)
PD5	OC1A (Timer/Counter1 Output compareA match output)
PD6	WR (Write strobe to external memory)
PD7	RD (Read strobe to external memory)

When the pins are used for the alternate function the DDRD and PORTD register has to be set according to the alternate function description.

THE PORT D DATA REGISTER - PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D DATA DIRECTION REGISTER - DDRD

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D INPUT PINS ADDRESS - PIND

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port D Input Pins address - PIND - is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

PORTD AS GENERAL DIGITAL I/O

PDn, General I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin the MOS pull up resistor is activated. To switch the pull up resistor off the PDn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 23. DDDn Bits on Port D Pins

DDDn	PORTDn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PDn will source current (IIL) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

ALTERNATE FUNCTIONS OF PORTD

$\overline{\text{RD}}$ - PORTD, Bit 7:
$\overline{\text{RD}}$ is the external data memory read control strobe.

$\overline{\text{WR}}$ - PORTD, Bit 6:
$\overline{\text{WR}}$ is the external data memory write control strobe.

OC1- PORTD, Bit 5:
OC1, Output compare match output: The PD5 pin can serve as an external output when the Timer/Counter1 compare matches. The PD5 pin has to be configured as an output (DDD5 set (one)) to serve this function. See the Timer/Counter1 description for further details, and how to enable the output. The OC1 pin is also the output pin for the PWM mode timer function.

<u>**INT1 - PORTD, Bit 3:**</u>
INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

<u>**INT0 - PORTD, Bit 2:**</u>
INT0, External Interrupt source 0: The PD2 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

<u>**TXD - PORTD, Bit 1:**</u>
Transmit Data (Data output pin for the UART). When the UART transmitter is enabled, this pin is configured as an output regardless of the value of DDRD1.

<u>**RXD - PORTD, Bit 0:**</u>
Receive Data (Data input pin for the UART). When the UART receiver is enabled this pin is configured as an output regardless of the value of DDRD0. When the UART forces this pin to be an input, a logical one in PORTD0 vill turn on the internal pull-up.

PORTD SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

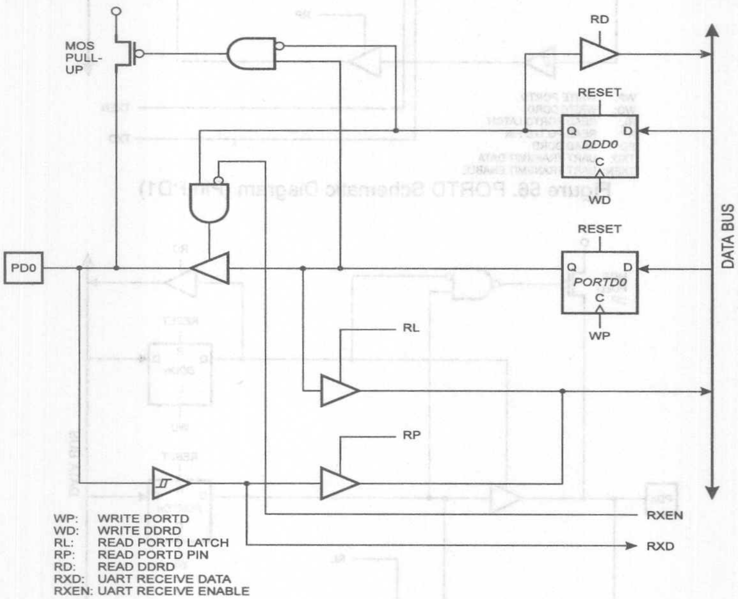


Figure 55. PORTD Schematic Diagram (Pin PD0)

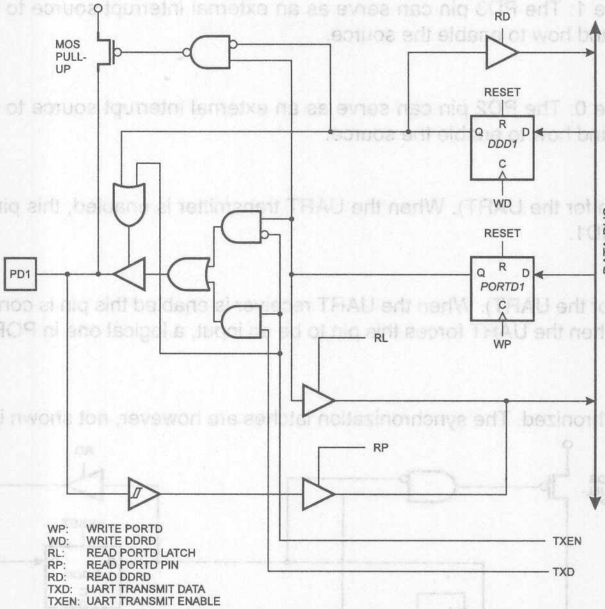


Figure 56. PORTD Schematic Diagram (Pin PD1)

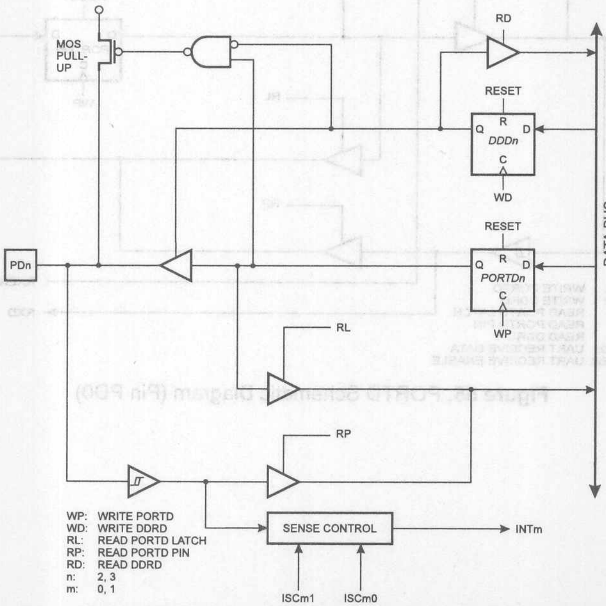


Figure 57. PORTD Schematic Diagram (Pins PD2 and PD3)

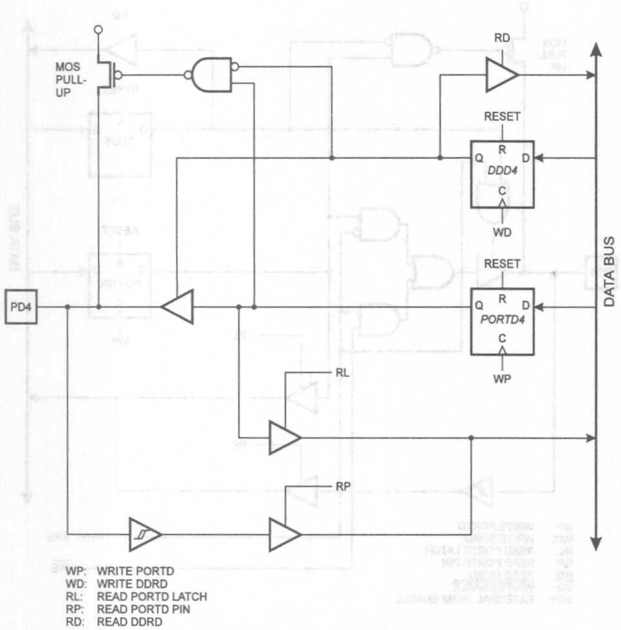


Figure 58. PORTD Schematic Diagram (Pin PD4)

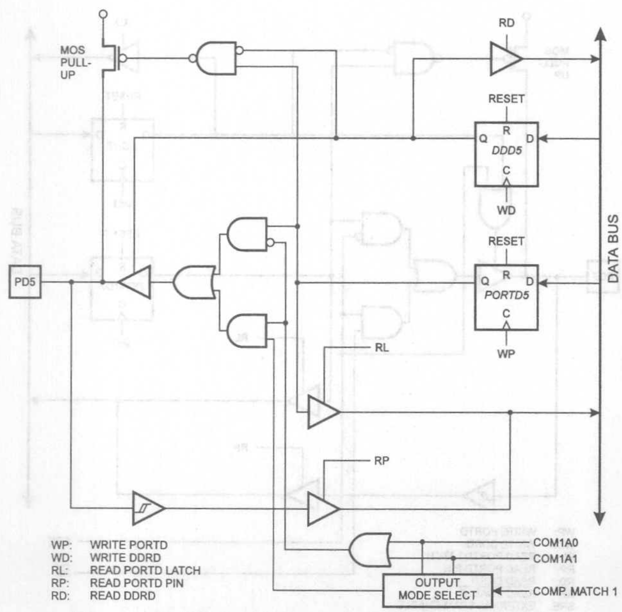


Figure 59. PORTD Schematic Diagram (Pin PD5)

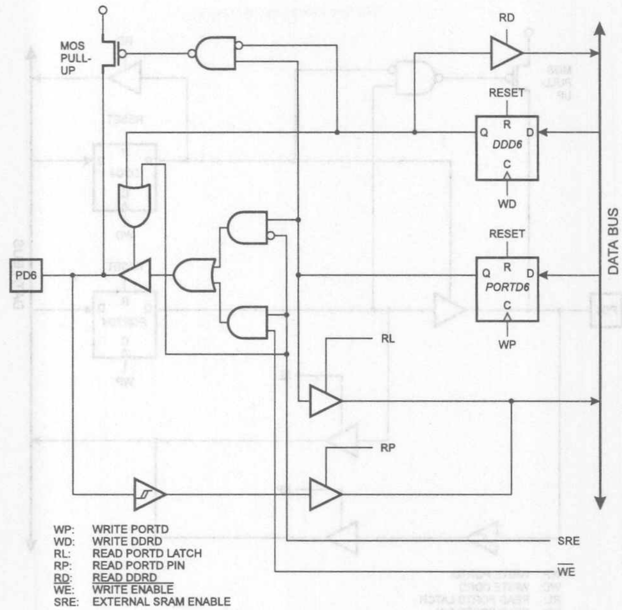


Figure 60. PORTD Schematic Diagram (Pin PD6)

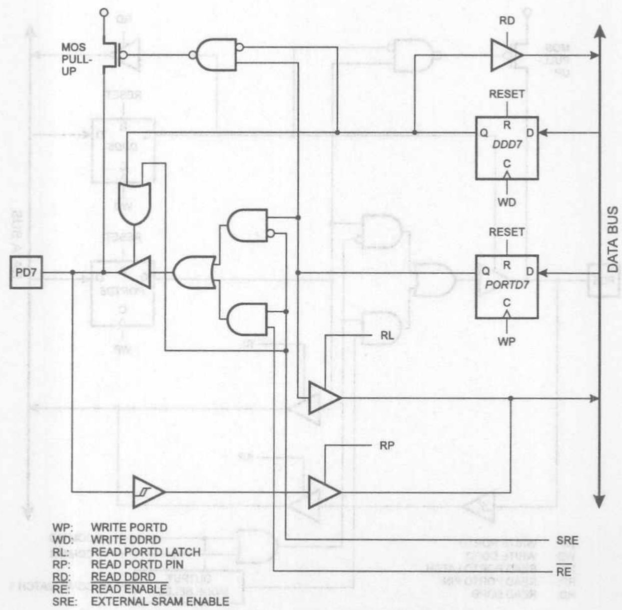


Figure 61. PORTD Schematic Diagram (Pin PD7)

Memory Programming

Program Memory Lock Bits

The AT90S4414 MCU provides two lock bits which can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table 24.

Table 24. Lock Bit Protection Modes

Program Lock Bits			Protection Type
Mode	LB1	LB2	
1	1	1	No program lock features
2	0	1	Further programming of the Flash is disabled
3	0	0	Same as mode 2, but verify is also disabled.

Note: The Lock Bits can only be erased with the Chip Erase operation.

Fuse Bits

The AT90S4414 has two fuse bits, SPIEN and FSTRT.

- When SPIEN is programmed ('0'), Serial Program Downloading is enabled. Default value is programmed ('0').
- When FSTRT is programmed ('0'), the short start-up time is selected. Default value is unprogrammed ('1'). Parts with this bit pre-programmed ('0') can be delivered on demand.

These bits are not accessible in Serial Programming Mode and are not affected by a chip erase.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode. The three bytes reside in a separate address space, and for the AT90S4414 they are:

1. \$000: \$1E (indicates manufactured by Atmel)
2. \$001: \$92 (indicates 4 kB Flash memory)
3. \$002: \$01 (indicates 90S4414 device when \$001 is \$92)

Programming the Flash and EEPROM

Atmel's AT90S4414 offers 4K bytes of in-system reprogrammable Flash Program memory and 256 bytes of EEPROM Data memory.

The AT90S4414 is normally shipped with the on-chip Flash Program and EEPROM Data memory arrays in the erased state (i.e. contents = \$FF) and ready to be programmed. This device supports a High-Voltage (12V) Parallel programming mode and a Low-Voltage Serial programming mode. The +12V is used for programming enable only, and no current of significance is drawn by this pin. The serial programming mode provides a convenient way to download the Program and Data into the AT90S4414 inside the user's system.

The Program and Data memory arrays on the AT90S4414 are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided with the self-timed programming operation in the serial programming mode.

Parallel Programming

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory + Program Memory Lock bits and Fuse bits in the AT90S4414.

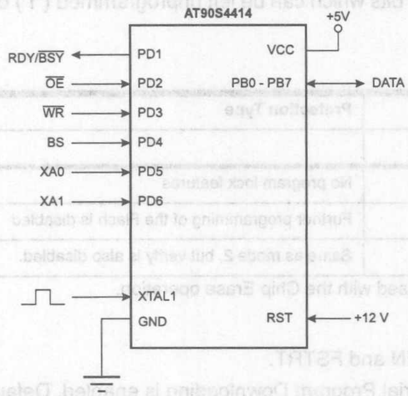


Figure 62. Parallel Programming

SIGNAL NAMES

In this section, some pins of the AT90S4414 are referenced by signal names describing their functionality during parallel programming rather than their pin names. Pins not described in the following table are referenced by pin names.

Table 25. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY / $\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active Low)
$\overline{WR}$	PD3	I	Write Pulse (Active Low)
BS	PD4	I	Byte Select
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1

The XA1/XA0 bits determine the action taken when the XTAL1 pin is given a positive pulse. The bit settings are shown in the following table:

Table 26. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or Low address byte for Flash determined by BS)
0	1	Load Data (High or Low data byte for Flash determined by BS)
1	0	Load Command
1	1	No Action, Idle

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action on input or output. The command is a byte where the different bits are assigned functions as shown in the following table:

Table 27. Command Byte Bit Coding

Bit#	Meaning when Set
7	Chip Erase
6	Write Fuse Bits. Located in the data byte at the following bit positions: D5: SPIEN Fuse, D0: FSTRT Fuse (Note: Write '0' to program, '1' to erase)
5	Write Lock Bits. Located in the data byte at the following bit positions: D1: LB1, D0: LB2 (Note: write '0' to program)
4	Write Flash or EEPROM (determined by bit 0)
3	Read signature row
2	Read Lock and Fuse Bits. Located in the data byte at the following bits positions: D7: LB1, D6: LB2, D5: SPIEN Fuse, D0: FSTRT Fuse (Note: '0' means programmed)
1	Read from Flash or EEPROM (determined by bit 0)
0	0 : Flash Access, 1 : EEPROM Access

ENTER PROGRAMMING MODE

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5 V between VCC and GND.
2. Set $\overline{RESET}$ and BS pins to '0' and wait at least 100 ns.
3. Apply 12V to and wait at least 100 ns before changing BS.

CHIP ERASE

The chip erase will erase the Flash and EEPROM memories plus Lock bits. The lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A chip erase must be performed before the chip is programmed.

Load Command "Chip Erase"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'.
3. Set PB(7:0) to '1000 0000'. This is the command for Chip erase.
4. Give XTAL1 a positive pulse. This loads the command, and starts the erase of the Flash and EEPROM arrays. After pulsing XTAL1, give $\overline{WR}$ a negative pulse to enable lock bit erase at the end of the erase cycle, then wait for at least 10 ms. Chip erase does not generate any activity on the RDY/$\overline{BSY}$ pin.

PROGRAMMING THE FLASH

Load Command "Program Flash"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'
3. Set PB(7:0) to '0001 0000'. This is the command for Flash programming.
4. Give XTAL1 a positive pulse. This loads the command.

Load Address Low byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '0'. This selects Low address.
3. Set PB(7:0) = Address Low byte ($00 - $FF)
4. Give XTAL1 a positive pulse. This loads the Address Low byte.

Load Address High byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '1'. This selects High address.
3. Set PB(7:0) = Address High byte ($00 - $07)
4. Give XTAL1 a positive pulse. This loads the Address High byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data Low byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data Low byte

1. Set BS to ('0').
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high to program the next byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data High byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data High byte

1. Set BS to '1'.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY / $\overline{BSY}$ goes low.
3. Wait until RDY / $\overline{BSY}$ goes high to program the next byte.

The loaded command and address are retained in the device during programming. To simplify programming, the following should be considered.

- The command for Flash programming needs only be loaded before programming of the first byte.
- Address High byte needs only be loaded before programming a new 256 word page in the Flash.

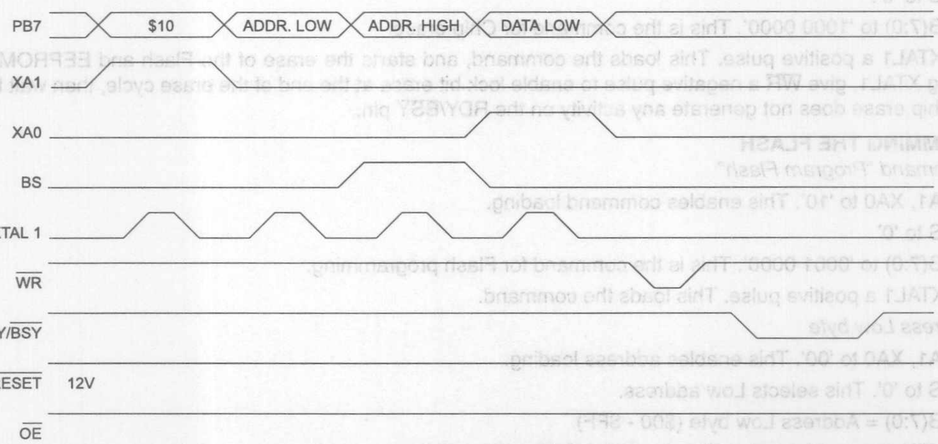


Figure 63. Programming Flash Low Byte

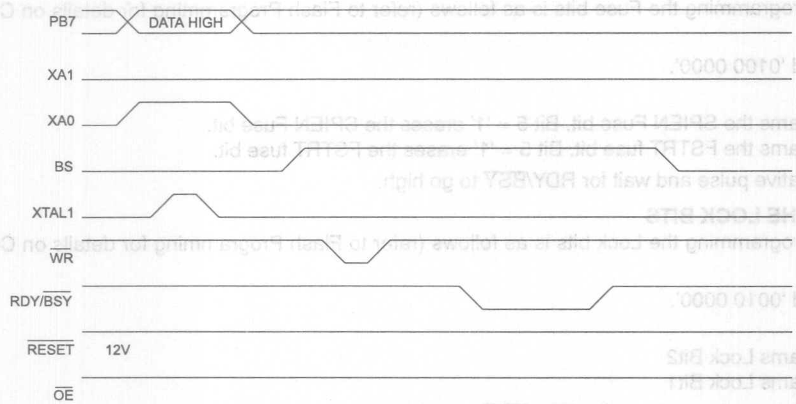


Figure 64. Programming Flash High Byte

PROGRAMMING THE EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0001 0001'.
2. Load Low EEPROM Address ($00 - $FF)
3. Load Low EEPROM Data ($00 - $FF)
4. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The Command needs only be loaded before programming the first byte.

READING THE FLASH

The algorithm for reading the Flash memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0010'.
2. Load Low Address ($00 - $FF)
3. Load High Address ($00 - $07)
4. Set $\overline{OE}$ to '0', and BS to '0'. The Low Data byte can now be read at PB(7:0)
5. Set BS to '1'. The High Data byte can now be read from PB(7:0)
6. Set $\overline{OE}$ to '1'.

The Command needs only be loaded before reading the first byte.

READING THE EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0011'.
2. Load Low Address ($00 - $FF)
3. Set $\overline{OE}$ to '0', and BS to '0'. The EEPROM Data byte can now be read at PB(7:0)
4. Set $\overline{OE}$ to '1'.

The Command needs only be loaded before reading the first byte.

PROGRAMMING THE FUSE BITS

The algorithm for programming the Fuse bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0100 0000'.
2. Load Data.
   Bit 5 = '0' programs the SPIEN Fuse bit. Bit 5 = '1' erases the SPIEN Fuse bit.
   Bit 0 = '0' programs the FSTRT fuse bit. Bit 5 = '1' erases the FSTRT fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

PROGRAMMING THE LOCK BITS

The algorithm for programming the Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0010 0000'.
2. Load Data.
   Bit 2 = '0' programs Lock Bit2
   Bit 1 = '0' programs Lock Bit1
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The lock bits can only be cleared by executing a chip erase.

READING THE FUSE AND LOCK BITS

The algorithm for reading the Fuse and Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0100'.
2. Set $\overline{OE}$ to '0', and BS to '1'. The Status of Fuse and Lock bits can now be read at PB(7:0)
   Bit 7: Lock Bit1 ('0' means programmed)
   Bit 6: Lock Bit2 ('0' means programmed)
   Bit 5: SPIEN Fuse ('0' means programmed, '1' means erased)
   Bit 0: FSTRT Fuse ('0' means programmed, '1' means erased)
3. Set $\overline{OE}$ to '1'.

Observe especially that BS needs to be set to '1'.

READING THE SIGNATURE BYTES

The algorithm for reading the Signature Bytes bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 1000'.
2. Load Low address ($00 - $02)
3. Set $\overline{OE}$ to '0', and BS to '0'. The Selected Signature byte can now be read at PB(7:0)
4. Set $\overline{OE}$ to '1'.

The command needs only be programmed before reading the first byte.

Serial Downloading

Both the Program and Data memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into $FF.

The Program and EEPROM memory arrays have separate address spaces:

$0000 to $07FF for Program memory and $0000 to $00FF for EEPROM memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2.The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 1 XTAL1 clock cycle

High: > 4 XTAL1 clock cycles

SERIAL PROGRAMMING ALGORITHM

To program and verify the AT90S4414 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 28):

1. *Power-up sequence:*

   Apply power between VCC and GND while $\overline{\text{RESET}}$ and SCK are set to '0'. (If the programmer can not guarantee that SCK is held low during power-up, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to '0'.) If a crystal is not connected across pins XTAL1 and XTAL2, apply a 0 to 20 MHz clock to the XTAL1 pin.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB5. Refer to the above section for minimum high and low periods for the serial clock input (SCK).
3. If a chip erase is performed (must be done to erase the Flash), wait 10ms, give $\overline{\text{RESET}}$ a positive pulse and start over again from Step 2.
4. The Flash or EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. The next byte can be written after 4 ms.
5. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.
6. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
7. Power-off sequence (if needed):

   Set XTAL1 to '0' (if a crystal is not used).

   Set $\overline{\text{RESET}}$ to '1'.

   Turn V_{CC} power off

Table 28. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both 4K & 256 byte memory arrays
Read Program Memory	0010 **H**000	xxxx **0aaa**	**bbbb bbbb**	**oooo oooo**	Read **H**(high or low) data **o** from Program memory at word address **a**:**b**
Write Program Memory	0100 **H**000	xxxx **0aaa**	**bbbb bbbb**	**iiii iiii**	Write **H**(high or low) data **i** to Program memory at word address **a**:**b**
Read EEPROM Memory	1010 0000	xxxx xxx0	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **b**
Write EEPROM Memory	1100 0000	xxxx xxx0	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **b**
Write Lock Bits	1010 1100	111x x**21**x	xxxx xxxx	xxxx xxxx	Write lock bits. Set bits **1**,**2**='0' to program lock bits.
Read Device Code	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Device Code **o** at address **b**

Notes: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1 - High Byte
o = data out
i = data in
x = don't care
1 = lock bit 1
2 = lock bit 2

Programming Characteristics

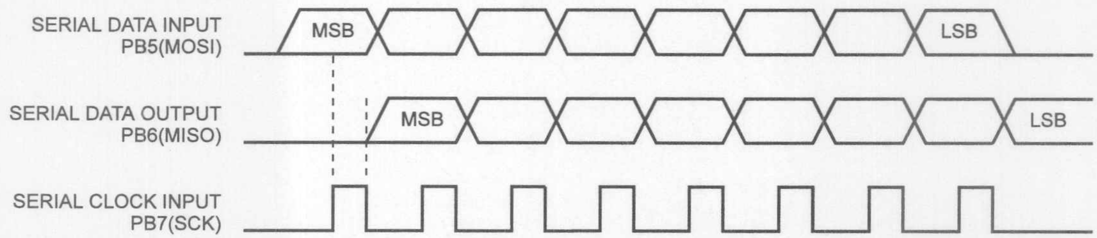


Figure 65. Serial Downloading Waveforms

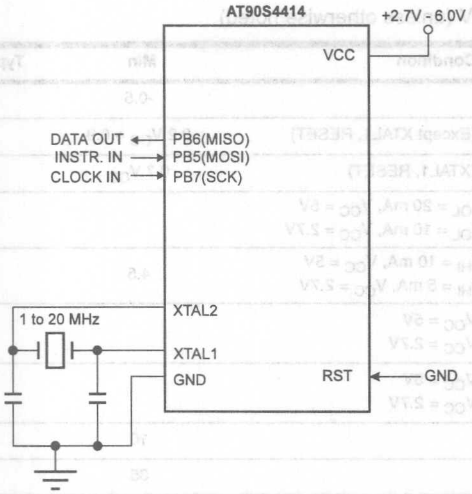


Figure 66. Serial Programming and Verify

When *writing* serial data to the AT90S4414, data is clocked on the *rising* edge of CLK.

When *reading* data from the AT90S4414, data is clocked on the *falling* edge of CLK. See Figure 65 for an explanation.

Absolute Maximum Ratings*

Operating Temperature	-40°C to +105°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
I/O Pin Maximum Current	40.0 mA
Maximum Current VCC and GND	140.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

4

DC Characteristics

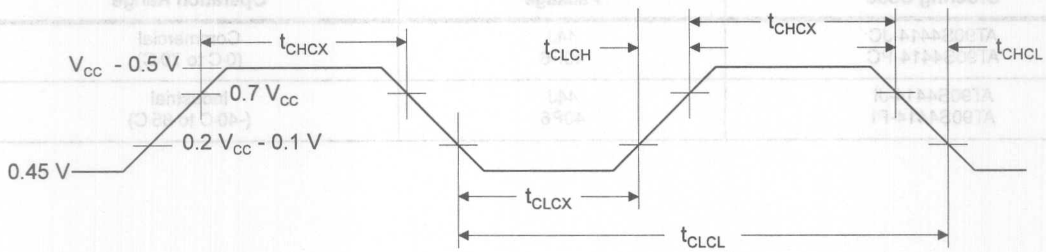
T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		0.2 Vcc - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RESET)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RESET)	0.7 V_{CC}		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage [(1)] (Ports B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V			0.5	V
V_{OH}	Output High Voltage (Ports B,C,D)	I_{HI} = 10 mA, V_{CC} = 5V I_{HI} = 5 mA, V_{CC} = 2.7V	4.5			V
I_{OH}	Output Source Current (Ports B,C,D)	V_{CC} = 5V V_{CC} = 2.7V			10 5	mA
I_{OL}	Output Sink Current (Port B,C,D)	V_{CC} = 5V V_{CC} = 2.7V			20 10	mA
RRST	Reset Pulldown Resistor		10		50	kΩ
$R_{I/O}$	I/O Pin Pull-Up Resistor		35		120	kΩ
I_{CC}	Power Supply Current	Active Mode, 3V, 4MHz		3.5		mA
		Idle Mode 3V, 4MHz		1000		µA
I_{CC}	Power Down Mode(2)	WDT enabled, 3V		50		µA
		WDT disabled, 3V		<1		µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V			20	mV
I_{ACLK}	Analog Comparator Input Leakage Current		1	5	10	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

Notes:

1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum total I_{OL} for all output pins: 80 mA
   Port A: 26 mA
   Ports A, B, D: 15 mA
   Maximum total I_{OL} for all output pins: 70 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification.
   Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	10	0	20	MHz
t_{CLCL}	Clock Period	100		41.7		ns
t_{CHCX}	High Time	40		16.7		ns
t_{CLCX}	Low Time	40		16.7		ns
t_{CLCH}	Rise Time		10		4.15	ns
t_{CHCL}	Fall Time		10		4.15	ns

Ordering Information

Ordering Code	Package	Operation Range
AT90S4414-JC AT90S4414-PC	44J 40P6	Commercial (0°C to 70°C)
AT90S4414-JI AT90S4414-PI	44J 40P6	Industrial (-40°C to 85°C)

Package Type	
44J	44 Lead, Plastic J-Leaded Chip Carrier (PLCC)
40P6	40 Lead, 0.600" Wide, Plastic Dual in Line Package (PDIP)

AT90S4414 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	4-23
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	4-24
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	4-24
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-	4-29
$3A ($5A)	GIFR	INTF1	INTF0							4-29
$39 ($59)	TIMSK	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-	4-29
$38 ($58)	TIFR	TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	-	4-30
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00	4-31
$34 ($54)	Reserved									
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	4-34
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								4-35
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	4-37
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	4-38
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte								4-39
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								4-39
$2B ($4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								4-40
$2A ($4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								4-40
$29 ($49)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								4-40
$28 ($48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								4-40
$27 ($47)	Reserved									
$26 ($46)	Reserved									
$25 ($45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								4-40
$24 ($44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								4-40
$23 ($43)	Reserved									
$22 ($42)	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	4-43
$20 ($40)	Reserved									
$1F ($3F)	EEARH	-	-	-	-	-	-	-	EEAR9	4-44
$1E ($3E)	EEARL	EEPROM Address Register Low								4-44
$1D ($3D)	EEDR	EEPROM Data Register								4-44
$1C ($3C)	EECR	-	-	-	-	-	EEMWE	EEWE	EERE	4-45
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	4-59
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	4-59
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	4-59
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	4-61
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	4-61
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	4-61
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	4-66
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	4-66
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	4-66
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	4-68
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	4-68
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	4-68
$0F ($2F)	SPDR	SPI Data Register								4-50
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-	4-49
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	4-49
$0C ($2C)	UDR	UART I/O Data Register								4-53
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-	4-53
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	4-54
$09 ($29)	UBRR	UART Baud Rate Register								4-56
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	4-57
...	Reserved									
$00 ($20)	Reserved									

4

AT90S4414 Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	3
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	3
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

4

Contents

5

5

Features

- Utilizes the AVR® Enhanced RISC Architecture
- AVR - High Performance and Low Power RISC Architecture
- 120 Powerful Instructions - Most Single Clock Cycle Execution
- 8K bytes of In-System Reprogrammable Downloadable Flash
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 512 bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 512 bytes Internal SRAM
- 32 x 8 General Purpose Working Registers
- 32 Programmable I/O Lines
- Programmable Serial UART
- SPI Serial Interface
- V_{CC}: 2.7 - 6.0V
- Fully Static Operation, 0 - 20 MHz
- Instruction Cycle Time: 50 ns @ 20 MHz
- One 8-Bit Timer/Counter with Separate Prescaler
- One 16-Bit Timer/Counter with Separate Prescaler and Compare and Capture Modes
- Dual PWM
- External and Internal Interrupt Sources
- Programmable Watchdog Timer with On-Chip Oscillator
- On-Chip Analog Comparator
- Low Power Idle and Power Down Modes
- Programming Lock for Software Security

8-Bit AVR® Microcontroller with 8K bytes Downloadable Flash

Preliminary

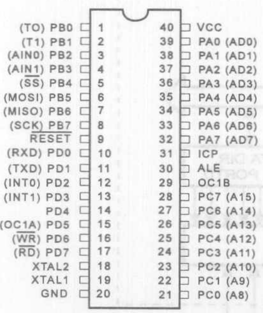
5

Description

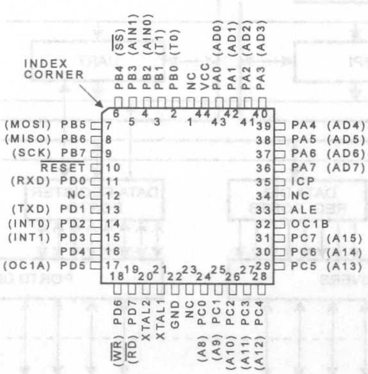
The AT90S8515 is a low-power CMOS 8-bit microcontroller based on the *AVR*® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S8515 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Pin Configurations

PDIP



PLCC



0841A

Block Diagram

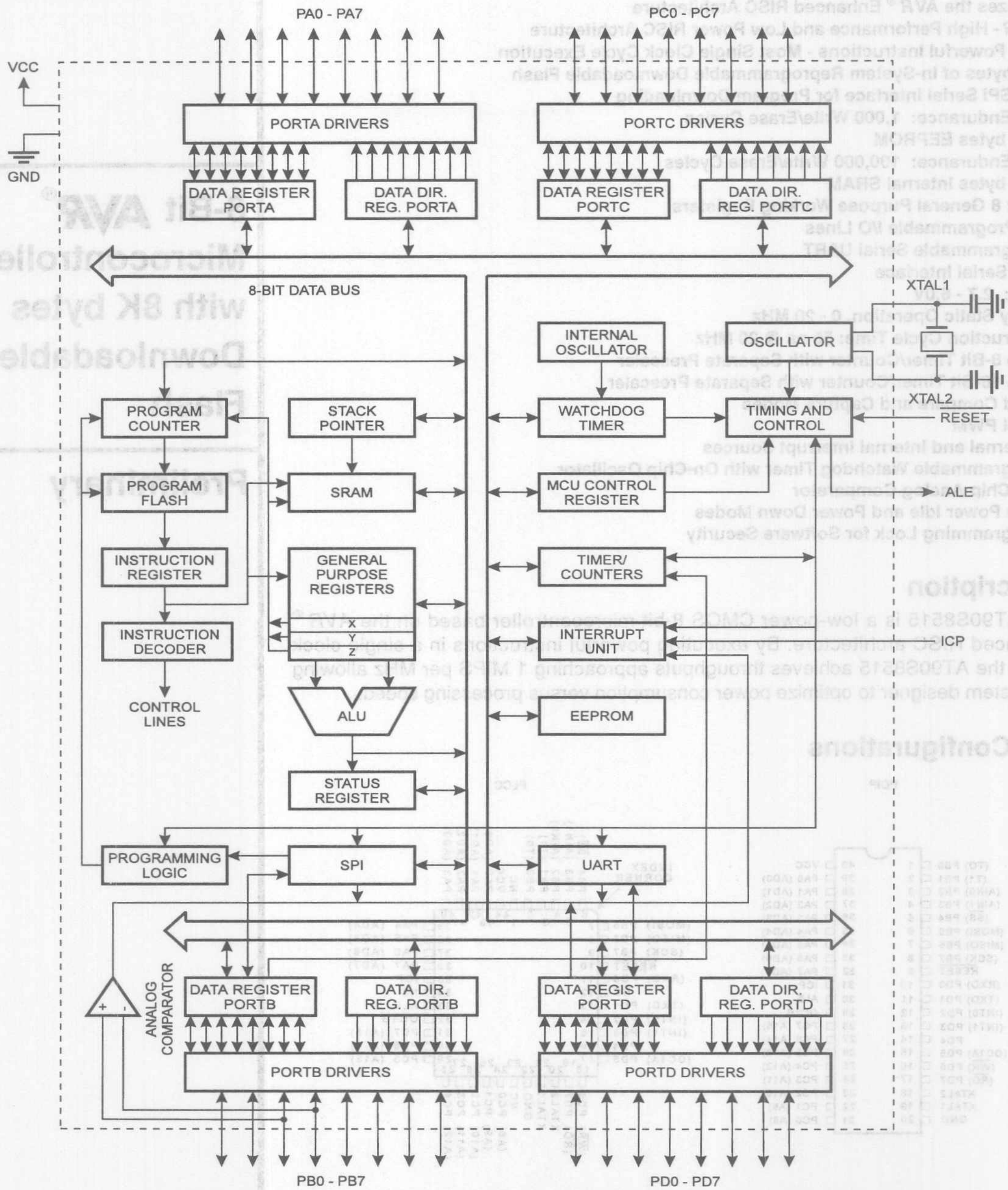


Figure 1. The AT90S8515 Block Diagram

Description (Continued)

The *AVR* core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The AT90S8515 provides the following features: 8K bytes of Downloadable Flash, 512 bytes EEPROM, 512 bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, flexible timer/counters with compare modes, internal and external interrupts, a programmable serial UART, programmable Watchdog Timer with internal oscillator, an SPI serial port and two software selectable power saving modes. The Idle Mode stops the CPU while allowing the SRAM, timer/counters, SPI port and interrupt system to continue functioning. The power down mode saves the register contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The device is manufactured using Atmel's high density non-volatile memory technology. The on-chip Downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining an enhanced RISC 8-bit CPU with Downloadable Flash on a monolithic chip, the Atmel AT90S8515 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The AT90S8515 *AVR* is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC

Supply voltage

GND

Ground

Port A (PA7..PA0)

Port A is an 8-bit bidirectional I/O port. Port pins can provide internal pullups (selected for each bit). The Port A output buffers can sink 20mA and can drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

Port A serves as Multiplexed Address/Data input/output when using external SRAM.

Port B (PB7..PB0)

Port B is an 8-bit bidirectional I/O pins with internal pullups. The Port B output buffers can sink 20 mA. As inputs, Port B pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port B also serves the functions of various special features of the AT90S8515 as listed on Page 5-62.

Port C (PC7..PC0)

Port C is an 8-bit bidirectional I/O port with internal pullups. The Port C output buffers can sink 20 mA. As inputs, Port C pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port C also serves as Address output when using external SRAM.

Port D (PD7..PD0)

Port D is an 8-bit bidirectional I/O port with internal pullups. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Port D also serves the functions of various special features of the AT90S8515 as listed on Page 5-68.

$\overline{\text{RESET}}$

Reset input. A low on this pin for two machine cycles while the oscillator is running resets the device.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier

ICP

ICP is the input pin for the Timer/Counter1 Input Capture function.

OC1B

OC1B is the output pin for the Timer/Counter1 Output CompareB function

ALE

ALE is the Address Latch Enable used when the External Memory is enabled. The ALE strobe is used to latch the low-order address (8 bits) into an address latch during the first access cycle, and the AD0-7 pins are used for data during the second access cycle.

Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 3.

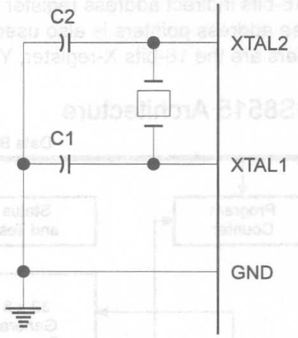


Figure 2. Oscillator Connections

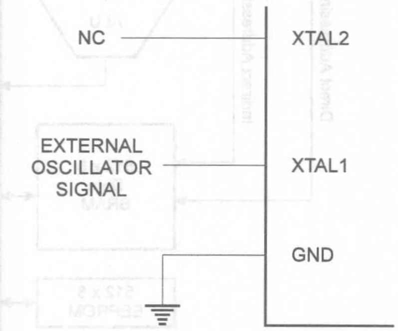


Figure 3. External Clock Drive Configuration

AT90S8515 *AVR* Enhanced RISC Microcontroller CPU

The AT90S8515 *AVR* RISC microcontroller is upward compatible with the *AVR* Enhanced RISC Architecture. The programs written for the AT90S8515 MCU are fully compatible with the range of *AVR* 8-bit MCUs (AT90Sxxxx) with respect to source code and clock cycles for execution.

Architectural Overview

The fast-access register file concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one ALU (Arithmetic Logic Unit) operation is executed. Two operands are output from the register file, the operation is executed, and the result is stored back in the register file - in one clock cycle.

Six of the 32 registers can be used as three 16-bits indirect address register pointers for Data Space addressing - enabling efficient address calculations. One of the three address pointers is also used as the address pointer for the constant table look up function. These added function registers are the 16-bits X-register, Y-register and Z-register.

AVR AT90S8515 Architecture

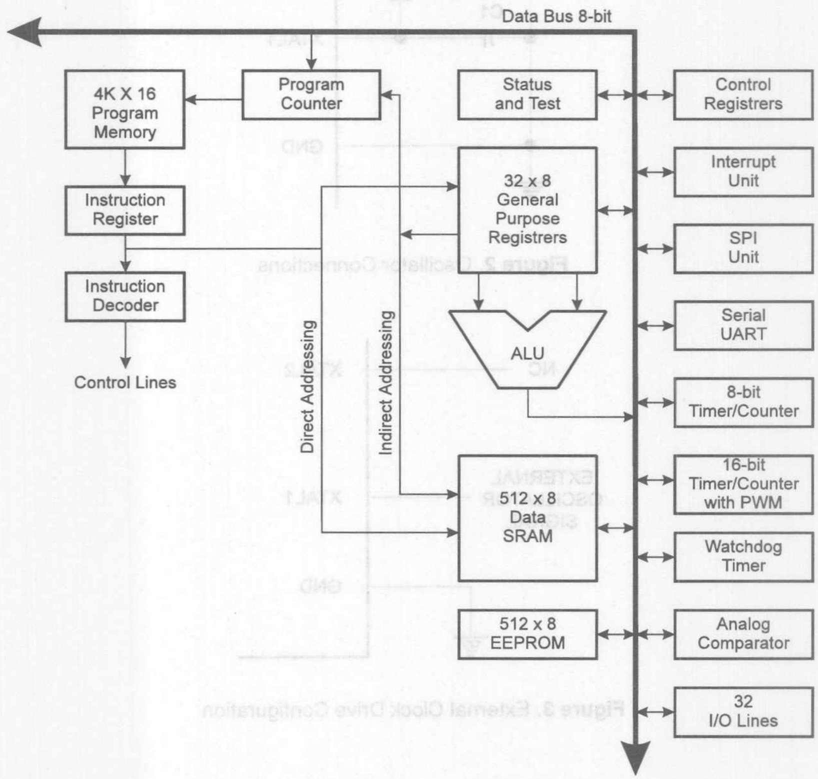


Figure 4. The AT90S8515 *AVR* Enhanced RISC Architecture

The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S8515 *AVR* Enhanced RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the register file as well. This is enabled by the fact that the register file is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, Timer/Counters, A/D-converters, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the register file, $20 - $5F.

The *AVR* uses a Harvard architecture concept - with separate memories and buses for program and data. The program memory is executed with a single level pipelining. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is in-system downloadable Flash memory.

With the relative jump and call instructions, the whole 4K address space is directly accessed. Most *AVR* instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

During interrupts and subroutine calls, the return address program counter (PC) is stored on the stack. The stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 16-bit stack pointer SP is read/write accessible in the I/O space.

The 512 bytes data SRAM can be easily accessed through the five different addressing modes supported in the *AVR* architecture.

The memory spaces in the *AVR* architecture are all linear and regular memory maps.

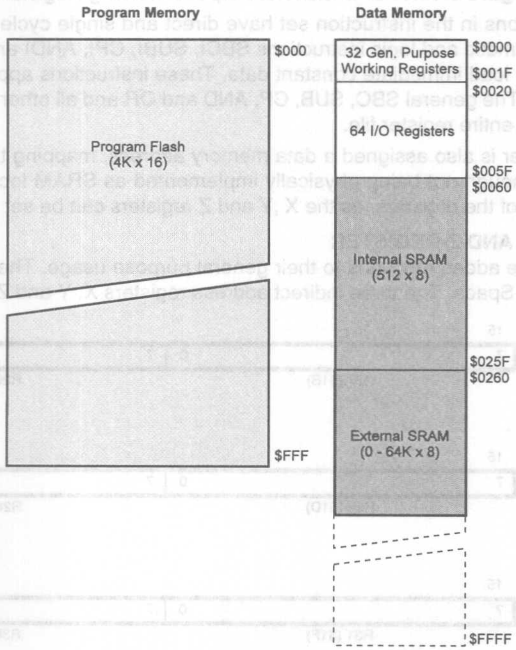


Figure 5. Memory Maps

A flexible interrupt module has its control registers in the I/O space with an additional global interrupt enable bit in the status register. All the different interrupts have a separate interrupt vector in the interrupt vector table at the beginning of the program memory. The different interrupts have priority in accordance with their interrupt vector position. The lower the interrupt address vector the higher priority.

The General Purpose Register File

Figure 6 shows the structure of the 32 general purpose working registers in the CPU.

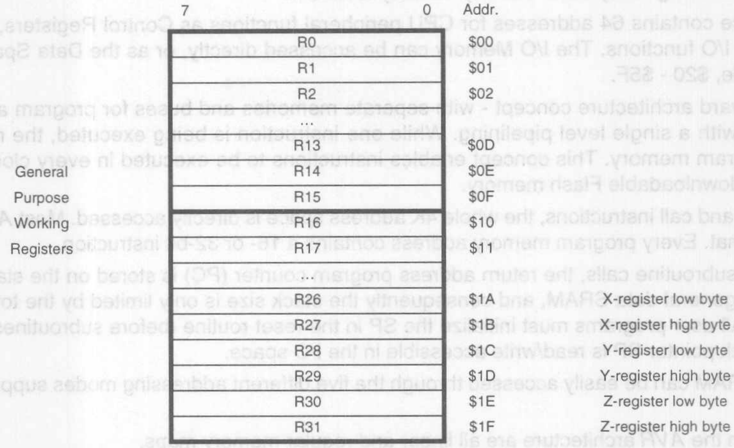


Figure 6. *AVR* CPU General Purpose Working Registers

All the register operating instructions in the instruction set have direct and single cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI and ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the register file - R16..R31. The general SBC, SUB, CP, AND and OR and all other operations between two registers or on a single register apply to the entire register file.

As shown in Figure 6, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although not being physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X ,Y and Z registers can be set to index any register in the file.

THE X-REGISTER, Y-REGISTER AND Z-REGISTER

The registers R26..R31 have some added functions to their general purpose usage. These registers are address pointers for indirect addressing of the Data Space. The three indirect address registers X, Y and Z are defined as:

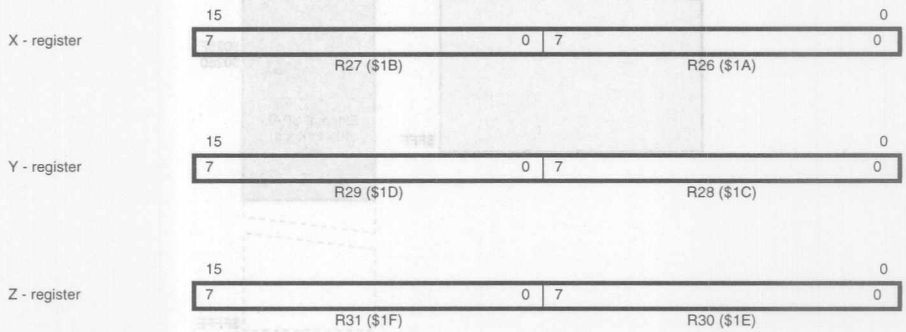


Figure 7. The X, Y and Z Registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement (see the descriptions for the different instructions).

The ALU - Arithmetic Logic Unit

The high-performance *AVR* ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the register file are executed. The ALU operations are divided into three main categories - arithmetic, logic and bit-functions. Some microcontrollers in the *AVR* product family feature a hardware multiplier in the arithmetic part of the ALU.

The Downloadable Flash Program Memory

The AT90S8515 contains 8K bytes on-chip downloadable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 4K x 16. The Flash memory has an endurance of at least 1000 write/erase cycles. The AT90S8515 Program Counter (PC) is 12 bits wide, thus addressing the 4096 program memory addresses.

See Page 5-78 for a detailed description on Flash data downloading.

Constant tables must be allocated within the address 0-4K (see the LPM - Load Program Memory instruction description).

See Page 5-14 for the different program memory addressing modes.

The SRAM Data Memory - Internal and External

The following figure shows how the AT90S8515 SRAM Memory is organized:

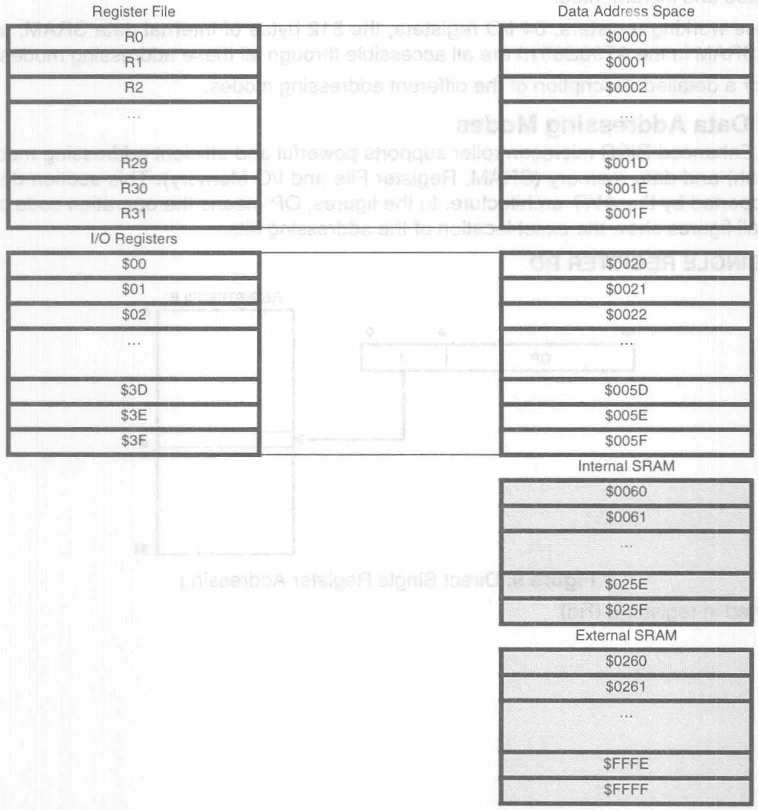


Figure 8. SRAM Organization

The lower 608 Data Memory locations address the Register file, the I/O Memory and the internal data SRAM. The first 96 locations address the Register File + I/O Memory, and the next 512 locations address the internal data SRAM. An optional external data SRAM can be placed in the same SRAM memory space. This SRAM will occupy the location following the internal SRAM and up to as much as 64K - 1, depending on SRAM size.

When the addresses accessing the data memory space exceeds the internal data SRAM locations, the external data SRAM is accessed using the same instructions as for the internal data SRAM access. When the internal data space is accessed, the read and write strobe pins (RD and $\overline{WR}$) are inactive during the whole access cycle. External SRAM operation is enabled by setting the SRE bit in the MCUCR register. See Page 5-31 for details.

Accessing external SRAM takes one additional clock cycle per byte compared to the internal SRAM. This applies to commands LD, ST, LDS, STS, PUSH, POP. If the stack is placed in the external SRAM, interrupts, subroutine calls and returns will require two more clock cycles. When the external SRAM is used with wait state, all external SRAM access takes four clock cycles extra.

The five different addressing modes for the data memory cover: Direct, Indirect with Displacement, Indirect, Indirect with Pre-Decrement and Indirect with Post-Increment. In the register file, registers R26 to R31 feature the indirect addressing pointer registers.

The direct addressing reaches the entire data space.

The Indirect with Displacement mode features a 63 address locations reach from the base address given by the Y or Z-register.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y and Z are decremented and incremented.

The 32 general purpose working registers, 64 I/O registers, the 512 bytes of internal data SRAM, and the 64K bytes of optional external data SRAM in the AT90S8515 are all accessible through all these addressing modes.

See the next section for a detailed description of the different addressing modes.

The Program and Data Addressing Modes

The AT90S8515 *AVR* Enhanced RISC microcontroller supports powerful and efficient addressing modes for access to the program memory (Flash) and data memory (SRAM, Register File and I/O Memory). This section describes the different addressing modes supported by the *AVR* architecture. In the figures, OP means the operation code part of the instruction word. To simplify, not all figures show the exact location of the addressing bits.

REGISTER DIRECT, SINGLE REGISTER RD

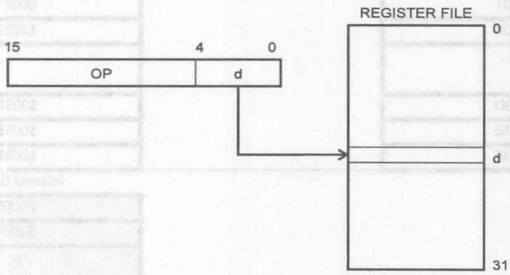


Figure 9. Direct Single Register Addressing

The operand is contained in register d (Rd).

REGISTER DIRECT, TWO REGISTERS RD AND RR

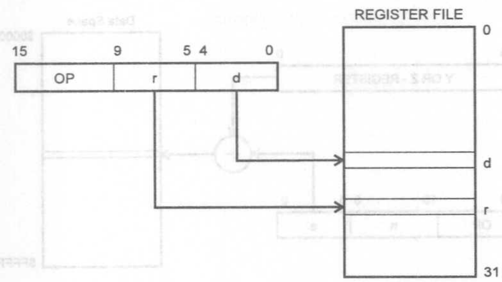


Figure 10. Direct Register Addressing, Two Registers

Operands are contained in register r (Rr) and d (Rd). The result is stored in register d (Rd).

I/O DIRECT

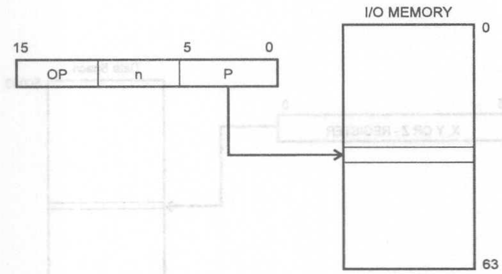


Figure 11. I/O Direct Addressing

Operand address is contained in 6 bits of the instruction word. n is the destination or source register address.

DATA DIRECT

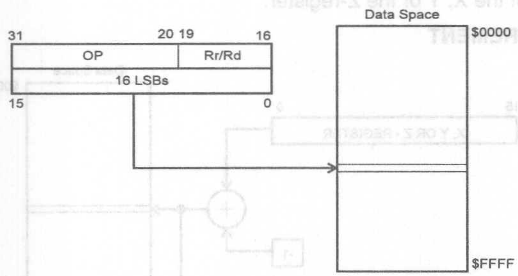


Figure 12. Direct Data Addressing

A 16-bit Data Address is contained in the 16 LSBs of a two-word instruction. Rd/Rr specify the destination or source register.

DATA INDIRECT WITH DISPLACEMENT

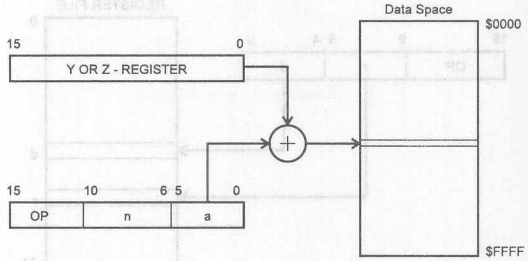


Figure 13. Data Indirect with Displacement

Operand address is the result of the Y or Z-register contents added to the address contained in 6 bits of the instruction word.

DATA INDIRECT

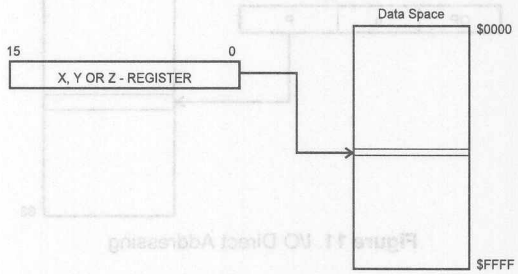


Figure 14. Data Indirect Addressing

Operand address is the contents of the X, Y or the Z-register.

DATA INDIRECT WITH PRE-DECREMENT

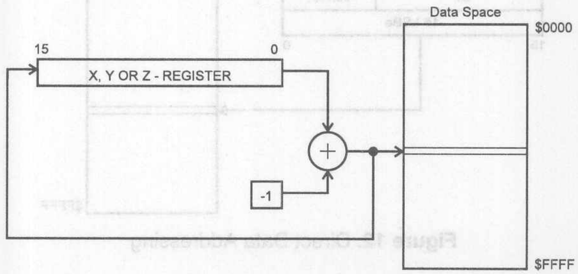


Figure 15. Data Indirect Addressing With Pre-Decrement

The X, Y or the Z-register is decremented before the operation. Operand address is the decremented contents of the X, Y or the Z-register.

DATA INDIRECT WITH POST-INCREMENT

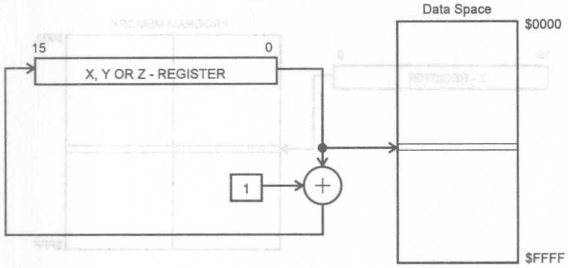


Figure 16. Data Indirect Addressing With Post-Increment

The X, Y or the Z-register is incremented after the operation. Operand address is the content of the X, Y or the Z-register prior to incrementing.

CONSTANT ADDRESSING USING THE LPM INSTRUCTION

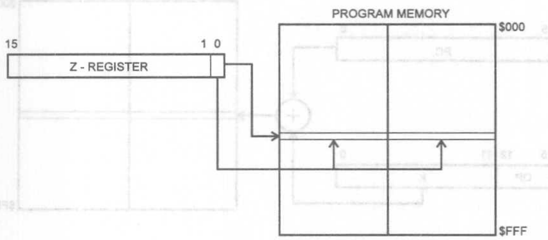


Figure 17. Code Memory Constant Addressing

Constant byte address is specified by the Z-register contents. The 15 MSBs select word address (0 - 4K) and LSB, select low byte if cleared (LSB = 0) or high byte if set (LSB = 1).

DIRECT PROGRAM ADDRESS, JMP AND CALL

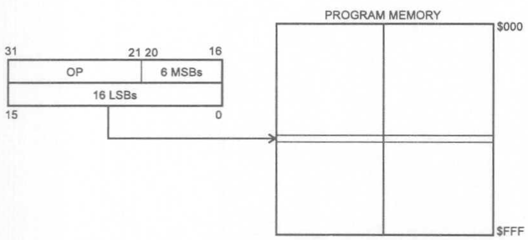


Figure 18. Direct Program Memory Addressing

Program execution continues at the address immediate in the instruction words.

INDIRECT PROGRAM ADDRESSING, IJMP AND ICALL

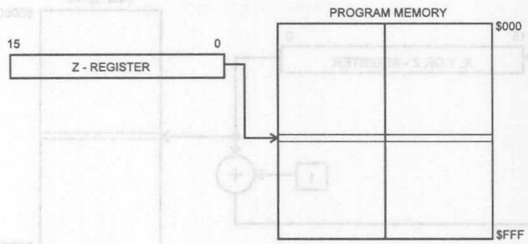


Figure 19. Indirect Program Memory Addressing

Program execution continues at address contained by the Z-register (i.e. the PC is loaded with the contents of the Z-register).

RELATIVE PROGRAM ADDRESSING, RJMP AND RCALL

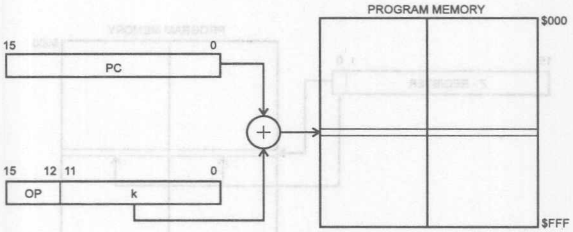


Figure 20. Relative Program Memory Addressing

Program execution continues at address PC + k. The relative address k is ± 2K.

The EEPROM Data Memory

The AT90S8515 contains 512 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on Page 5-44 specifying the EEPROM address registers, the EEPROM data register, and the EEPROM control register.

For the SPI data downloading, see Page 5-78 for a detailed description.

Memory Access Times and Instruction Execution Timing

This section describes the general access timing concepts for instruction execution and internal memory access.

The *AVR* CPU is driven by the System Clock Ø, directly generated from the external clock crystal for the chip. No internal clock division is used.

Figure 21 shows the parallel instruction fetches and instruction executions enabled by the Harvard architecture and the fast-access register file concept. This is the basic pipelining concept to obtain up to 1 MIPS per MHz with the corresponding unique results for functions per cost, functions per clocks, and functions per power-unit.

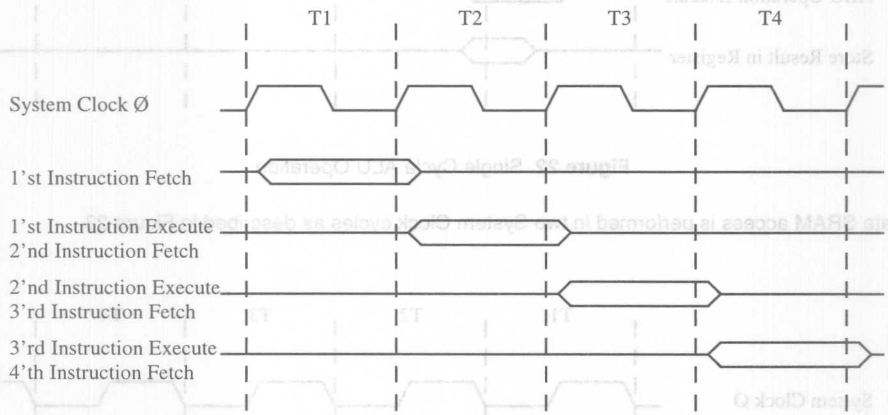


Figure 21. The Parallel Instruction Fetches and Instruction Executions

Figure 22 shows the internal timing concept for the register file. In a single clock cycle an ALU operation using two register operands is executed, and the result is stored back to the destination register.

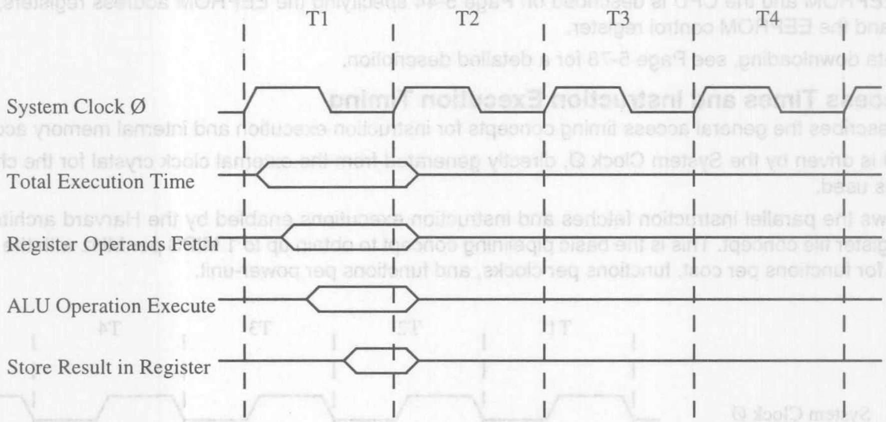


Figure 22. Single Cycle ALU Operation

The internal data SRAM access is performed in two System Clock cycles as described in Figure 23.

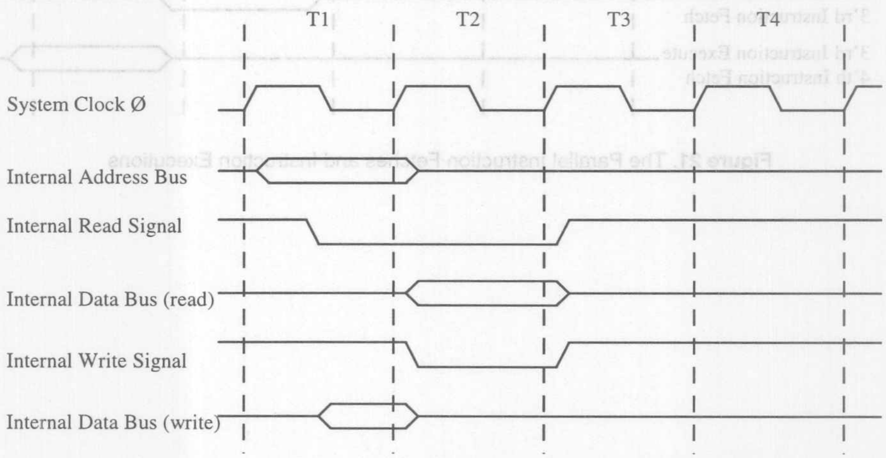


Figure 23. On-Chip Data SRAM Access Cycles

The external data SRAM access is performed in two System Clock cycles as described in Figure 23.

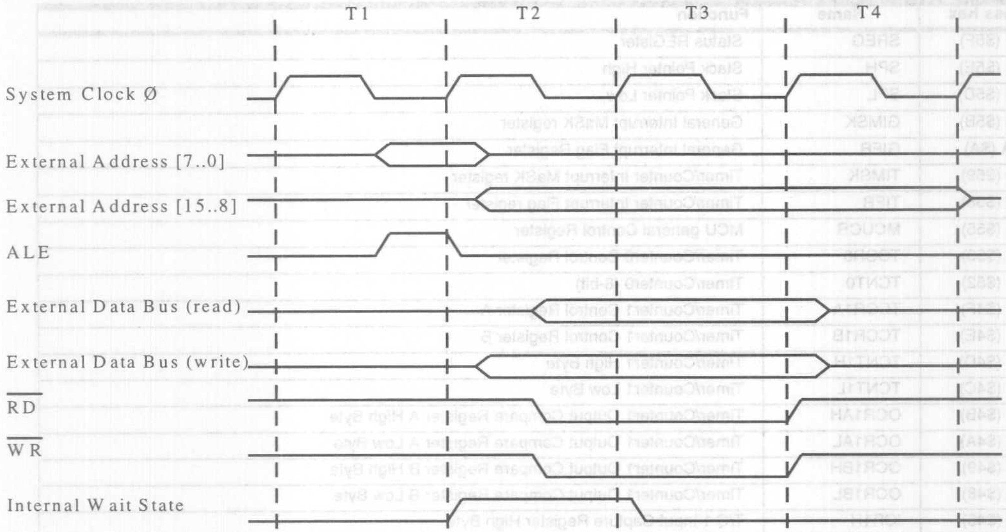


Figure 24. External Data SRAM Memory Cycles without Wait State

The external data SRAM memory access cycle with the Wait State bit enabled (Wait State active) is shown in Figure 25.

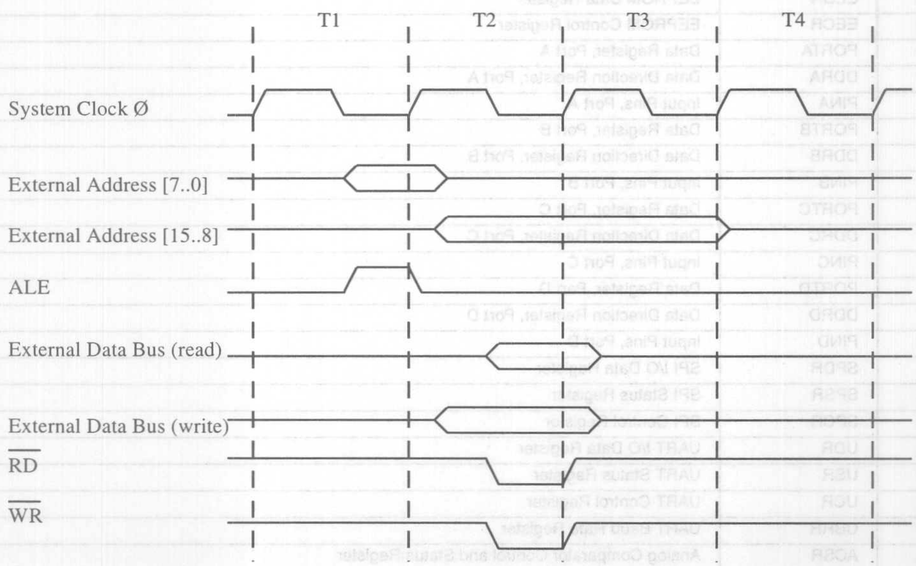


Figure 25. External Data SRAM Memory Cycles with Wait State

I/O Memory

The I/O space definition of the AT90S8515 is shown in the following table:

Table 1. AT90S8515 I/O Space

Address Hex	Name	Function
$3F ($5F)	SREG	Status REGister
$3E ($5E)	SPH	Stack Pointer High
$3D ($5D)	SPL	Stack Pointer Low
$3B ($5B)	GIMSK	General Interrupt MaSK register
$3A ($A)	GIFR	General Interrupt Flag Register
$39 ($59)	TIMSK	Timer/Counter Interrupt MaSK register
$38 ($58)	TIFR	Timer/Counter Interrupt Flag register
$35 ($55)	MCUCR	MCU general Control Register
$33 ($53)	TCCR0	Timer/Counter0 Control Register
$32 ($52)	TCNT0	Timer/Counter0 (8-bit)
$2F ($4F)	TCCR1A	Timer/Counter1 Control Register A
$2E ($4E)	TCCR1B	Timer/Counter1 Control Register B
$2D ($4D)	TCNT1H	Timer/Counter1 High Byte
$2C ($4C)	TCNT1L	Timer/Counter1 Low Byte
$2B ($4B)	OCR1AH	Timer/Counter1 Output Compare Register A High Byte
$2A ($4A)	OCR1AL	Timer/Counter1 Output Compare Register A Low Byte
$29 ($49)	OCR1BH	Timer/Counter1 Output Compare Register B High Byte
$28 ($48)	OCR1BL	Timer/Counter1 Output Compare Register B Low Byte
$25 ($45)	ICR1H	T/C 1 Input Capture Register High Byte
$24 ($44)	ICR1L	T/C 1 Input Capture Register Low Byte
$21 ($41)	WDTCR	Watchdog Timer Control Register
$1F ($3E)	EEARH	EEPROM Address Register High Byte
$1E ($3E)	EEARL	EEPROM Address Register Low Byte
$1D ($3D)	EEDR	EEPROM Data Register
$1C ($3C)	EECR	EEPROM Control Register
$1B ($3B)	PORTA	Data Register, Port A
$1A ($3A)	DDRA	Data Direction Register, Port A
$19 ($39)	PINA	Input Pins, Port A
$18 ($38)	PORTB	Data Register, Port B
$17 ($37)	DDRB	Data Direction Register, Port B
$16 ($36)	PINB	Input Pins, Port B
$15 ($35)	PORTC	Data Register, Port C
$14 ($34)	DDRC	Data Direction Register, Port C
$13 ($33)	PINC	Input Pins, Port C
$12 ($32)	PORTD	Data Register, Port D
$11 ($31)	DDRD	Data Direction Register, Port D
$10 ($30)	PIND	Input Pins, Port D
$0F ($2F)	SPDR	SPI I/O Data Register
$0E ($2E)	SPSR	SPI Status Register
$0D ($2D)	SPCR	SPI Control Register
$0C ($2C)	UDR	UART I/O Data Register
$0B ($2B)	USR	UART Status Register
$0A ($2A)	UCR	UART Control Register
$09 ($29)	UBRR	UART Baud Rate Register
$08 ($28)	ACSR	Analog Comparator Control and Status Register

Note: reserved and unused locations are not shown in the table

All the different AT90S8515 I/Os and peripherals are placed in the I/O space. The different I/O locations are accessed by the IN and OUT instructions transferring data between the 32 general purpose working registers and the I/O space. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set chapter for more details.

When using the I/O specific commands, IN, OUT,SBIS and SBIC, the I/O addresses $00 - $3F must be used. When addressing I/O registers as SRAM, $20 must be added to this address. All I/O register addresses throughout this document are shown with the SRAM address in parentheses.

The different I/O and peripherals control registers are explained in the following chapters.

THE STATUS REGISTER - SREG

The *AVR* status register - SREG - at I/O space location $3F ($5F) is defined as:

Bit	7	6	5	4	3	2	1	0	
$3F ($5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - I : Global Interrupt Enable:

The global interrupt enable bit must be set (one) for the interrupts to be enabled. The individual interrupt enable control is then performed in the interrupt mask registers - GIMSK and TIMSK. If the global interrupt enable register is cleared (zero), none of the interrupts are enabled independent of the GIMSK and TIMSK values. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts.

Bit 6 - T : Bit Copy Storage:

The bit copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T bit as source and destination for the operated bit. A bit from a register in the register file can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the register file by the BLD instruction.

Bit 5 - H : Half Carry Flag:

The half carry flag H indicates a half carry in some arithmetic operations. See the Instruction Set Description for detailed information.

Bit 4 - S : Sign Bit, $S = N \oplus V$:

The S-bit is always an exclusive or between the negative flag N and the two's complement overflow flag V. See the Instruction Set Description for detailed information.

Bit 3 - V : Two's Complement Overflow Flag:

The two's complement overflow flag V supports two's complement arithmetics. See the Instruction Set Description for detailed information.

Bit 2 - N : Negative Flag:

The negative flag N indicates a negative result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 1 - Z : Zero Flag:

The zero flag Z indicates a zero result after the different arithmetic and logic operations. See the Instruction Set Description for detailed information.

Bit 0 - C : Carry Flag:

The carry flag C indicates a carry in an arithmetic or logic operation. See the Instruction Set Description for detailed information.

THE STACK POINTER - SP

The general *AVR* 16-bit Stack Pointer is effectively built up of two 8-bit registers in the I/O space locations $3E ($5E) and $3D ($5D). As the AT90S8515 supports up to 64 kB external SRAM, all 16-bits are used.

Bit	15	14	13	12	11	10	9	8	
$3E ($5E)	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
$3D ($5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The Stack Pointer points to the data SRAM stack area where the Subroutine and Interrupt Stacks are located. This Stack space in the data SRAM must be defined by the program before any subroutine calls are executed or interrupts are enabled. The Stack Pointer is decremented by one when data is pushed onto the Stack with the PUSH instruction, and it is decremented by two when data is pushed onto the Stack with subroutine CALL and interrupt. The Stack Pointer is incremented by one when data is popped from the Stack with the POP instruction, and it is incremented by two when data is popped from the Stack with return from subroutine RET or return from interrupt IRET.

Reset and Interrupt Handling

The AT90S8515 provides 12 different interrupt sources. These interrupts and the separate reset vector, each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be set (one) together with the I-bit in the status register in order to enable the interrupt.

The lowest addresses in the program memory space are automatically defined as the Reset and Interrupt vectors. The complete list of vectors is shown in Table 2. The list also determines the priority levels of the different interrupts. The lower the address the higher is the priority level. RESET has the highest priority, and next is INT0 - the External Interrupt Request 0 etc.

Table 2. Reset and Interrupt Vectors

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	Hardware Pin and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0, OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	UART, RX	UART, Rx Complete
11	$00A	UART, UDRE	UART Data Register Empty
12	$00B	UART, TX	UART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator

The most typical and general program setup for the Reset and Interrupt Vector Addresses are:

```
Address    Labels     Code                    Comments
$000                  rjmp    RESET         ; Reset Handle
$001                  rjmp    EXT_INT0      ; IRQ0 Handle
$002                  rjmp    EXT_INT1      ; IRQ1 Handle
$003                  rjmp    TIM1_CAPT     ; Timer1 capture Handle
$004                  rjmp    TIM1_COMPA    ; Timer1 compareA Handle
$005                  rjmp    TIM1_OVF      ; Timer1 overflow Handle
$006                  rjmp    TIM1_OVF      ; Timer1 overflow Handle
$007                  rjmp    TIM0_OVF      ; Timer0 overflow Handle
$008                  rjmp    SPI_HANDLE    ; SPI TX Handle
$009                  rjmp    UART_RXC      ; UART RX Complete Handle
$00a                  rjmp    UART_DRE      ; UDR Empty Handle
$00b                  rjmp    UART_TXC      ; UART TX Complete Handle
$00c                  rjmp    ANA_COMP      ; Analog Comparator Handle
;
$00d       MAIN:      <instr>  xxx          ; Main program start
…          …          …
```

RESET SOURCES

The AT90S8515 has three sources of reset:

- Power-On Reset. The MCU is reset when a supply voltage is applied to the VCC and GND pins.
- External Reset. The MCU is reset when a low level is present on the RESET pin for more than two XTAL cycles
- Watchdog Reset. The MCU is reset when the Watchdog timer period expires and the Watchdog is enabled.

During reset, all I/O registers are then set to their initial values, and the program starts execution from address $000. The instruction placed in address $000 must be an RJMP - relative jump - instruction to the reset handling routine. If the program never enables an interrupt source, the interrupt vectors are not used, and regular program code can be placed at these locations. The circuit diagram in Figure 26 shows the reset logic. Table 3 defines the timing and electrical parameters of the reset circuitry.

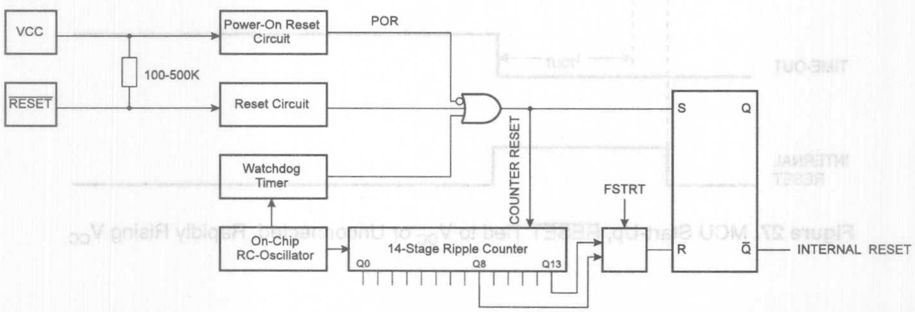


Figure 26. Reset Logic

Table 3. Reset Characteristics (V_{CC} = 5.0V)

Symbol	Parameter	Min	Typ	Max	Units
V_{POT}	Power-On Reset Threshold Voltage	1.8	2	2.2	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage		$V_{CC}/2$		V
t_{POR}	Power-On Reset Period	2	3	4	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Unprogrammed	11	16	21	ms
t_{TOUT}	Reset Delay Time-Out Period FSTRT Programmed	1.0	1.1	1.2	ms

POWER-ON RESET

A Power-On Reset (POR) circuit ensures that the device is not started until V_{CC} has reached a safe level. As shown in Figure 26, an internal timer clocked from the Watchdog timer oscillator prevents the MCU from starting until after a certain period after V_{CC} has reached the Power-On Threshold voltage - V_{POT}, regardless of the V_{CC} rise time (see Figure 27 and Figure 28). The total reset period is the Power-On Reset period - t_{POR} + the Delay Time-out period - t_{TOUT}. The FSTRT fuse bit in the Flash can be programmed to give a shorter start-up time if a ceramic resonator or any other fast-start oscillator is used to clock the MCU.

As the pin is pulled high by an on-chip resistor, the pin can be left unconnected if no external reset is required. Connecting to V_{CC} will have the same effect. By holding the pin low for a period after V_{CC} has been applied, the Power-On Reset period can be extended. Refer to Figure 29 for a timing example on this.

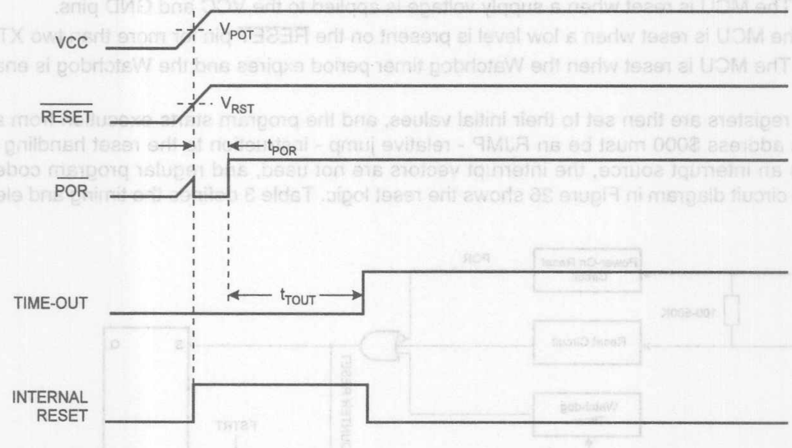


Figure 27. MCU Start-Up, $\overline{\text{RESET}}$ Tied to V_{CC} or Unconnected. Rapidly Rising V_{CC}

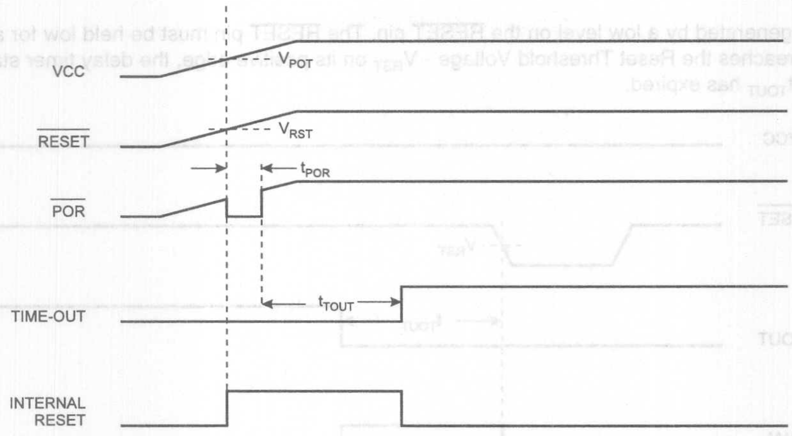


Figure 28. MCU Start-Up, RESET Tied to V_{CC} or Unconnected. Slowly Rising V_{CC}

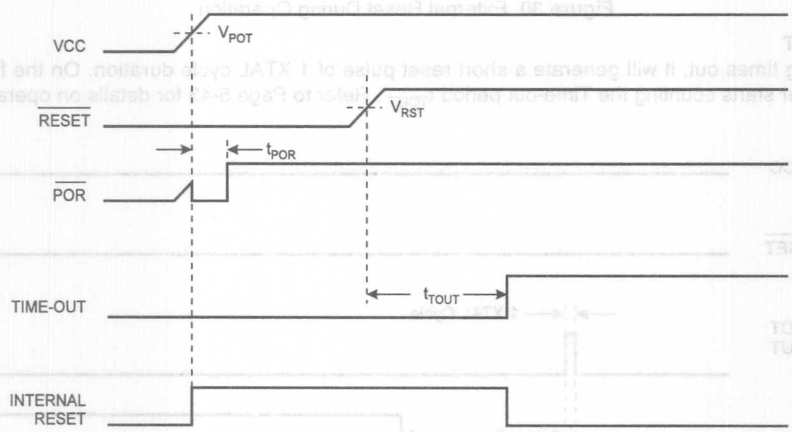


Figure 29. MCU Start-Up, RESET Controlled Externally

EXTERNAL RESET

An external reset is generated by a low level on the $\overline{RESET}$ pin. The $\overline{RESET}$ pin must be held low for at least two crystal clock cycles. When reaches the Reset Threshold Voltage - V_{RST} on its positive edge, the delay timer starts the MCU after the Time-out period t_{TOUT} has expired.

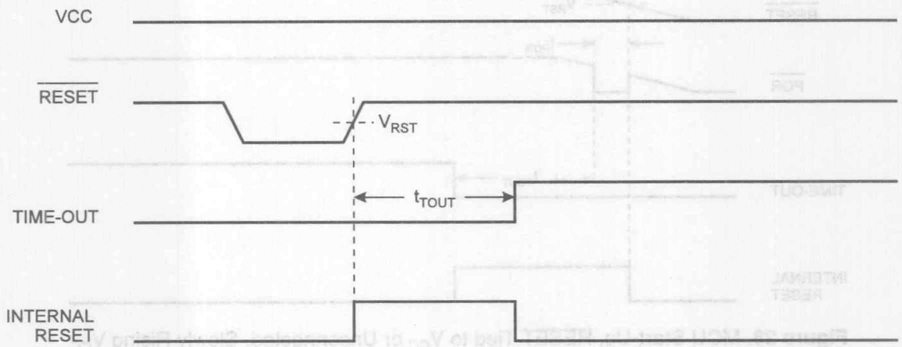


Figure 30. External Reset During Operation

WATCHDOG RESET

When the Watchdog times out, it will generate a short reset pulse of 1 XTAL cycle duration. On the falling edge of this pulse, the delay timer starts counting the Time-out period t_{TOUT} . Refer to Page 5-43 for details on operation of the Watchdog.

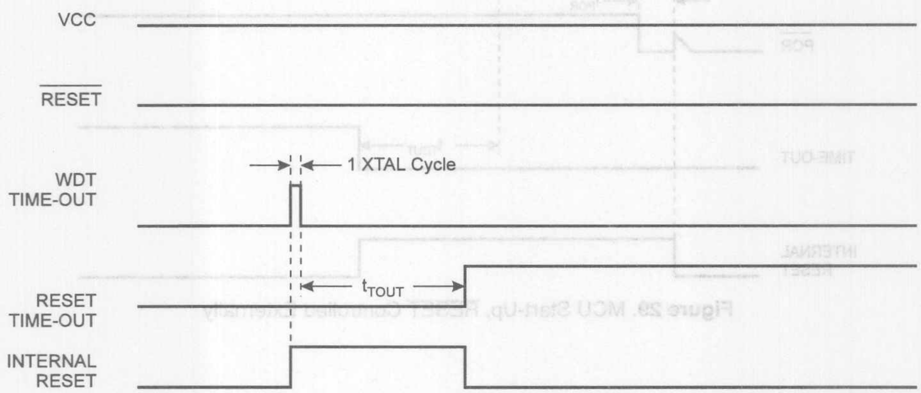


Figure 31. Watchdog Reset During Operation

INTERRUPT HANDLING

The AT90S8515 has two 8-bit Interrupt Mask control registers; GIMSK - General Interrupt Mask register and TIMSK - Timer/Counter Interrupt Mask register.

When an interrupt occurs, the Global Interrupt Enable I-bit is cleared (zero) and all interrupts are disabled. The user software must set (one) the I-bit to enable interrupts.

When the Program Counter is vectored to the actual interrupt vector in order to execute the interrupt handling routine, hardware clears the corresponding flag that generated the interrupt. Some of the interrupt flags can also be cleared by writing a logic one to the flag bit position(s) to be cleared.

THE GENERAL INTERRUPT MASK REGISTER - GIMSK

Bit	7	6	5	4	3	2	1	0	
$3B ($5B)	INT1	INT0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INT1 : External Interrupt Request 1 Enable:
When the INT1 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control1 bits 1/0 (ISC11 and ISC10) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT1 pin or level sensed. Activity on the pin will cause an interrupt request even if INT1 is configured as an output. The corresponding interrupt of External Interrupt Request 1 is executed from program memory address $002. See also "external Interrupts".

Bit 6 - INT0 : External Interrupt Request 0 Enable:
When the INT0 bit is set (one) and the I-bit in the Status Register (SREG) is set (one), the external pin interrupt is activated. The Interrupt Sense Control0 bits 1/0 (ISC01 and ISC00) in the MCU general Control Register (MCUCR) defines whether the external interrupt is activated on rising or falling edge of the INT0 pin or level sensed. Activity on the pin will cause an interrupt request even if INT0 is configured as an output. The corresponding interrupt of External Interrupt Request 0 is executed from program memory address $001. See also "External Interrupts."

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and always read as zero.

THE GENERAL INTERRUPT FLAG REGISTER - GIFR

Bit	7	6	5	4	3	2	1	0	
$3A ($5A)	INTF1	INTF0	-	-	-	-	-	-	GIMSK
Read/Write	R/W	R/W	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - INTF1 : External Interrupt Flag1:
When an event on the INT1 pin triggers an interrupt request, INTF1 becomes set (one). If the I-bit in SREG and the INT1 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $002. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bit 6 - INTF0 : External Interrupt Flag0:
When an event on the INT0 pin triggers an interrupt request, INTF0 becomes set (one). If the I-bit in SREG and the INT0 bit in GIMSK are set (one), the MCU will jump to the interrupt vector at address $001. The flag is cleared when the interrupt routine is executed. Alternatively, the flag can be cleared by writing a logical one to it.

Bits 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and always read as zero.

THE TIMER/COUNTER INTERRUPT MASK REGISTER - TIMSK

Bit	7	6	5	4	3	2	1	0	
$39 ($59)	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-	TIMSK
Read/Write	R/W	R/W	R/W	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOIE1 : Timer/Counter1 Overflow Interrupt Enable:
When the TOIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Overflow interrupt is enabled. The corresponding interrupt (at vector $006) is executed if an overflow in Timer/Counter1 occurs. The Overflow Flag (Timer/Counter1) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR. When Timer/Counter1 is in PWM mode, the Timer Overflow flag is set when the counter changes counting direction at $0000.

Bit 6 - OCE1A :Timer/Counter1 Output CompareA Match Interrupt Enable:
When the OCIE1A bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareA Match interrupt is enabled. The corresponding interrupt (at vector $004) is executed if a CompareA match in Timer/Counter1 occurs. The CompareA Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 5 - OCIE1B :Timer/Counter1 Output CompareB Match Interrupt Enable:
When the OCIE1B bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 CompareB Match interrupt is enabled. The corresponding interrupt (at vector $005) is executed if a CompareB match in Timer/Counter1 occurs. The CompareB Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 4 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

Bit 3 - TICIE1 : Timer/Counter1 Input Capture Interrupt Enable:
When the TICIE1 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter1 Input Capture Event Interrupt is enabled. The corresponding interrupt (at vector $003) is executed if a capture-triggering event occurs on pin 31, ICP. The Input Capture Flag in Timer/Counter1 is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

Bit 1 - TOIE0 : Timer/Counter0 Overflow Interrupt Enable:
When the TOIE0 bit is set (one) and the I-bit in the Status Register is set (one), the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt (at vector $008) is executed if an overflow in Timer/Counter0 occurs. The Overflow Flag (Timer0) is set (one) in the Timer/Counter Interrupt Flag Register - TIFR.

Bit 0 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

THE TIMER/COUNTER INTERRUPT FLAG REGISTER - TIFR

Bit	7	6	5	4	3	2	1	0	
$38 ($58)	TOV1	OCF1A	OCIFB	-	ICF1	-	TOV0	-	TIFR
Read/Write	R/W	R/W	R/W	R	R/W	R	R/W	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - TOV1 : Timer/Counter1 Overflow Flag:
The TOV1 is set (one) when an overflow occurs in Timer/Counter1. TOV1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV1 is cleared by writing a logic one to the flag. When the I-bit in SREG, and TOIE1 (Timer/Counter1 Overflow Interrupt Enable), and TOV1 are set (one), the Timer/Counter1 Overflow Interrupt is executed. In PWM mode, this bit is set when Timer/Counter1 changes counting direction at $0000.

Bit 6 - OCF1A : Output Compare Flag 1A:
The OCF1A bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1A - Output Compare Register 1A. OCF1A is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1A is cleared by writing a logic one to the flag. When the I-bit in SREG, and OCIE1A (Timer/Counter1 Compare match InterruptA Enable), and the OCF1A are set (one), the Timer/Counter1 Compare match Interrupt is executed.

Bit 5 - OCF1B : Output Compare Flag 1B:
The OCF1B bit is set (one) when compare match occurs between the Timer/Counter1 and the data in OCR1B - Output Compare Register 1B. OCF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, OCF1 is cleared by writing a logic one to the flag.. When the I-bit in SREG, and OCIE1B (Timer/Counter1 Compare match InterruptB Enable), and the OCF1B are set (one), the Timer/Counter1 Compare match Interrupt is executed.

Bit 4 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

Bit 3 - ICF1 : - Input Capture Flag 1:
The ICF1 bit is set (one) to flag an input capture event, indicating that the Timer/Counter1 value has been transferred to the input capture register - ICR1. ICF1 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ICF1 is cleared by writing a logic one to the flag.

Bit 2 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

Bit 1 - TOV0 : Timer/Counter0 Overflow Flag:
The bit TOV0 is set (one) when an overflow occurs in Timer/Counter0. TOV0 is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, TOV0 is cleared by writing a logic one to the flag. When the SREG I-bit, and TOIE0 (Timer/Counter0 Overflow Interrupt Enable), and TOV0 are set (one), the Timer/Counter0 Overflow interrupt is executed.

Bit 0 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and always reads zero.

EXTERNAL INTERRUPTS

The external interrupts are triggered by the INT1 and INT0 pins.Observe that, if enabled, the interrupts will trigger even if the INT0/INT1 pins are configured as outputs. This feature provides a way of generating a software interrupt. The external interrupts can be triggered by a falling or rising edge or a low level. This is set up as indicated in the specification for the MCU Control Register - MCUCR. When the external interrupt is enabled and is configured as level triggered, the interrupt will trigger as long as the pin is held low.

The external interrupts are set up as described in the specification for the MCU Control Register - MCUCR.

INTERRUPT RESPONSE TIME

The interrupt execution response for all the enabled *AVR* interrupts is 4 clock cycles minimum. After the 4 clock cycles the program vector address for the actual interrupt handling routine is executed. During this 4 clock cycle period, the Program Counter (2 bytes) is pushed onto the Stack, and the Stack Pointer is decremented by 2. The vector is a relative jump to the interrupt routine, and this jump takes 2 clock cycles. If an interrupt occurs during execution of a multi-cycle instruction, this instruction is completed before the interrupt is served.

A return from an interrupt handling routine (same as for a subroutine call routine) takes 4 clock cycles. During these 4 clock cycles, the Program Counter (2 bytes) is popped back from the Stack, and the Stack Pointer is incremented by 2. When *AVR* exits from an interrupt, it will always return to the main program and execute one more instruction before any pending interrupt is served.

Note that the Status Register - SREG - is not handled by the *AVR* hardware, neither for interrupts nor for subroutines. For the interrupt handling routines requiring a storage of the SREG, this must be performed by user software.

For Interrupts triggered by events that can remain static (E.g. the Output Compare Register1 A matching the value of Timer/Counter1) the interrupt flag is set when the event occurs. If the interrupt flag is cleared and the interrupt condition persists, the flag will not be set until the event occurs the next time.

MCU CONTROL REGISTER - MCUCR

The MCU Control Register contains control bits for general MCU functions.

Bit	7	6	5	4	3	2	1	0	
$35 ($55)	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - SRE : External SRAM Enable:
When the SRE bit is set (one), the external data SRAM is enabled, and the pin functions AD0-7 (Port A), A8-15 (Port C), $\overline{\text{WR}}$ and $\overline{\text{RD}}$ (Port D) are activated as the alternate pin functions. Then the SRE bit overrides any pin direction settings in the respective data direction registers. See "The SRAM Data Memory - Internal and External" for description of the External SRAM pin functions. When the SRE bit is cleared (zero), the external data SRAM is disabled, and the normal pin and data direction settings are used.

Bit 6 - SRW : External SRAM Wait State:
When the SRW bit is set (one), a one cycle wait state is inserted in the external data SRAM access cycle. When the SRW bit is cleared (zero), the external data SRAM access is executed with the normal two cycle scheme. See Figure 24 External Data SRAM Memory Cycles without Wait State and Figure 25: External Data SRAM Memory Cycles with Wait State.

Bit 5 - SE : Sleep Enable:
The SE bit must be set (one) to make the MCU enter the sleep mode when the SLEEP instruction is executed. To avoid the MCU entering the sleep mode unless it is the programmers purpose, it is recommended to set the Sleep Enable SE bit just before the execution of the SLEEP instruction.

Bit 4 - SM : Sleep Mode:
This bit selects between the two available sleep modes. When SM is cleared (zero), Idle Mode is selected as Sleep Mode. When SM is set (one), Power Down mode is selected as sleep mode. For details, refer to the paragraph "Sleep Modes" below.

Bit 3, 2 - ISC11, ISC10 : Interrupt Sense Control 1 bit 1 and bit 0:
The External Interrupt 1 is activated by the external pin INT1 if the SREG I-flag and the corresponding interrupt mask in the GIMSK is set. The level and edges on the external INT1 pin that activate the interrupt are defined in the following table:

Table 4. Interrupt 1 Sense Control

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Note: When changing the ISC11/ISC10 bits, INT1 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bit 1, 0 - ISC01, ISC00 : Interrupt Sense Control 0 bit 1 and bit 0:
The External Interrupt 0 is activated by the external pin INT0 if the SREG I-flag and the corresponding interrupt mask is set. The level and edges on the external INT0 pin that activate the interrupt are defined in the following table:

Table 5. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Reserved
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Note: When changing the ISC10/ISC00 bits, INT0 must be disabled by clearing its Interrupt Enable bit in the GIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Sleep Modes

To enter the sleep modes, the SE bit in MCUCR must be set (one) and a SLEEP instruction must be executed. If an enabled interrupt occurs while the MCU is in a sleep mode, the MCU awakes, executes the interrupt routine, and resumes execution from the instruction following SLEEP. The contents of the register file, SRAM and I/O memory are unaltered. If a reset occurs during sleep mode, the MCU wakes up and executes from the Reset vector.

Note that if a *level* triggered interrupt is used for wake-up from power down, the low level must be held for a time longer than the oscillator start-up time of 16 ms. Otherwise, the interrupt flag may return to zero before the MCU starts executing.

IDLE MODE

When the SM bit is cleared (zero), the SLEEP instruction forces the MCU into the Idle Mode stopping the CPU but allowing Timer/Counters, Watchdog and the interrupt system to continue operating. This enables the MCU to wake up from external triggered interrupts as well as internal ones like Timer Overflow interrupt and watchdog reset. If wakeup from the Analog Comparator interrupt is not required, the analog comparator can be powered down by setting the ACD-bit in the Analog Comparator Control and Status register - ACSR. This will reduce power consumption during Idle Mode.

POWER DOWN MODE

When the SM bit is set (one), the SLEEP instruction forces the MCU into the Power Down Mode. In this mode, the external oscillator is stopped. The user can select whether the watchdog shall be enabled during power-down mode. If the watchdog is enabled, it will wake up the MCU when the Watchdog Time-out period expires. If the watchdog is disabled, only an external reset or an external level triggered interrupt can wake up the MCU.

Timer / Counters

The AT90S8515 provides two general purpose Timer/Counters - one 8-bit T/C and one 16-bit T/C. The Timer/Counters have individual prescaling selection from the same 10-bit prescaling timer. Both Timer/Counters can either be used as a timer with an internal clock timebase or as a counter with an external pin connection which triggers the counting.

The Timer/Counter Prescaler

Figure 32 shows the general Timer/Counter prescaler.

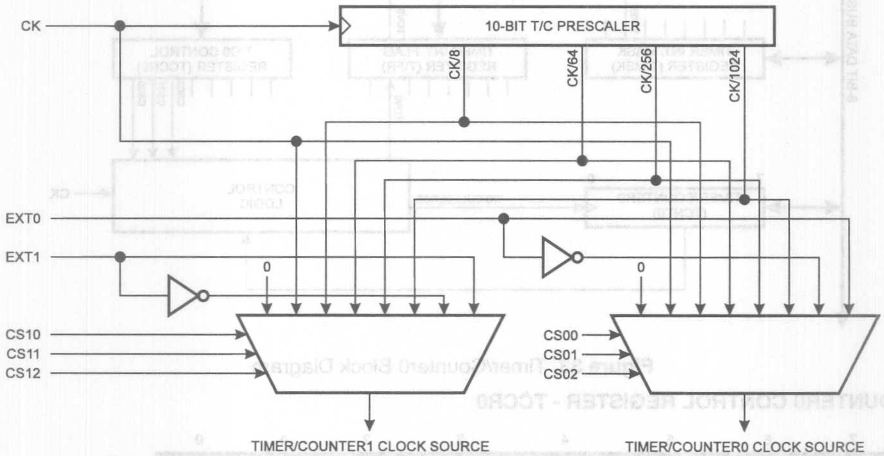


Figure 32. Timer/Counter Prescaler

The four different prescaled selections are: CK/8, CK/64, CK/256 and CK/1024 where CK is the oscillator clock. For the two Timer/Counters, added selections as CK, external source and stop, can be selected as clock sources.

The 8-Bit Timer/Counter0

Figure 33 shows the block diagram for Timer/Counter0.

The 8-bit Timer/Counter0 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter0 Control Register - TCCR0. The overflow status flag is found in the Timer/Counter Insterrupt Flag Register - TIFR. Control signals are found in the Timer/Counter0 Control Register - TCCR0. The interrupt enable/disable settings for Timer/Counter0 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter0 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 8-bit Timer/Counter0 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities make the Timer/Counter0 useful for lower speed functions or exact timing functions with infrequent actions.

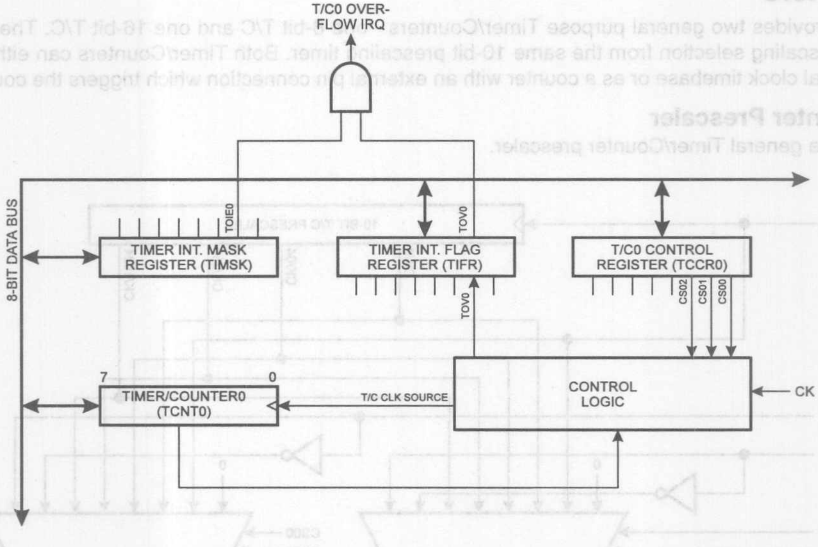


Figure 33. Timer/Counter0 Block Diagram

THE TIMER/COUNTER0 CONTROL REGISTER - TCCR0

Bit	7	6	5	4	3	2	1	0	
$33 ($53)	-	-	-	-	-	CS02	CS01	CS00	TCCR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7,6 - Res : Reserved bits:

These bits are reserved bits in the AT90S8515 and always read zero.

Bits 2,1,0 - CS02, CS01, CS00 : Clock Select0, bit 2,1 and 0:

The Clock Select0 bits 2,1 and 0 define the prescaling source of Timer0.

Table 6. Clock 0 Prescale Select

CS02	CS01	CS00	Description
0	0	0	Stop, the Timer/Counter0 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T0, falling edge
1	1	1	External Pin T0, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual data direction control register (cleared to zero gives an input pin).

THE TIMER COUNTER 0 - TCNT0

Bit	7	6	5	4	3	2	1	0	
\$32 (\$52)	MSB							LSB	TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The Timer/Counter0 is realized as an up-counter with read and write access. If the Timer/Counter0 is written and a clock source is present, the Timer/Counter0 continues counting in the clock cycle following the write operation.

The 16-Bit Timer/Counter1

Figure 34 shows the block diagram for Timer/Counter1.

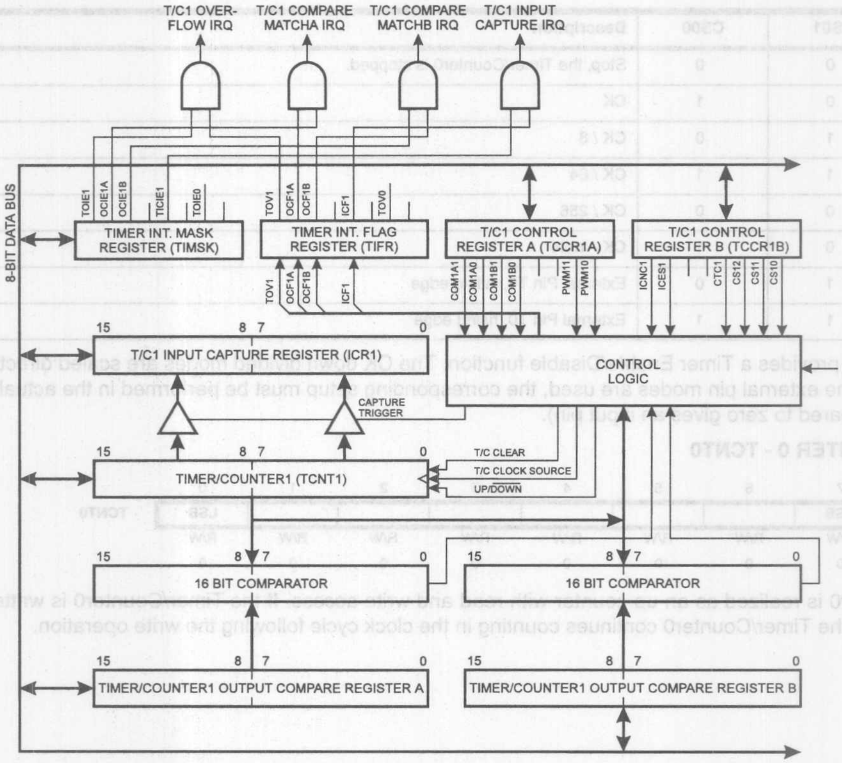


Figure 34. Timer/Counter1 Block Diagram

The 16-bit Timer/Counter1 can select clock source from CK, prescaled CK, or an external pin. In addition it can be stopped as described in the specification for the Timer/Counter1 Control Registers - TCCR1A and TCCR1B. The different status flags (overflow, compare match and capture event) and control signals are found in the Timer/Counter1 Control Registers - TCCR1A and TCCR1B. The interrupt enable/disable settings for Timer/Counter1 are found in the Timer/Counter Interrupt Mask Register - TIMSK.

When Timer/Counter1 is externally clocked, the external signal is synchronized with the oscillator frequency of the CPU. To assure proper sampling of the external clock, the minimum time between two external clock transitions must be at least one internal CPU clock period. The external clock signal is sampled on the rising edge of the internal CPU clock.

The 16-bit Timer/Counter1 features both a high resolution and a high accuracy usage with the lower prescaling opportunities. Similarly, the high prescaling opportunities makes the Timer/Counter1 useful for lower speed functions or exact timing functions with infrequent actions.

The Timer/Counter1 supports two Output Compare functions using the Output Compare Register 1 A and B - OCR1A and OCR1B as the data sources to be compared to the Timer/Counter1 contents. The Output Compare functions include optional clearing of the counter on compareA match, and actions on the Output Compare pins on both compare matches.

Timer/Counter1 can also be used as a 8, 9 or 10-bit Pulse With Modulator. In this mode the counter and the OCR1A/OCR1B registers serve as a dual glitch-free stand-alone PWM with centered pulses. Refer to Page 5-41 for a detailed description on this function.

The Input Capture function of Timer/Counter1 provides a capture of the Timer/Counter1 contents to the Input Capture Register - ICR1, triggered by an external event on the Input Capture Pin - ICP. The actual capture event settings are defined by the Timer/Counter1 Control Register - TCCR1B. In addition, the Analog Comparator can be set to trigger the Input Capture. Refer to the section, “The Analog Comparator”, for details on this. The ICP pin logic is shown in Figure 35.

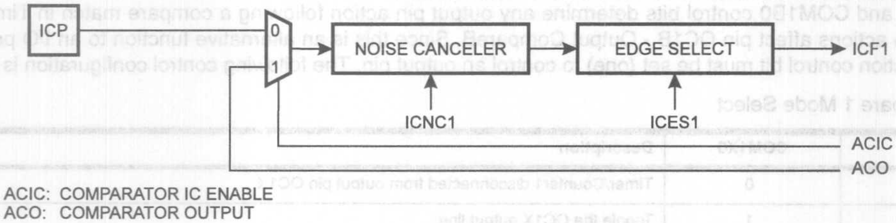


Figure 35. ICP Pin Schematic Diagram

The Timer/Counter1 input capture noise canceler block diagram is shown in Figure 36.

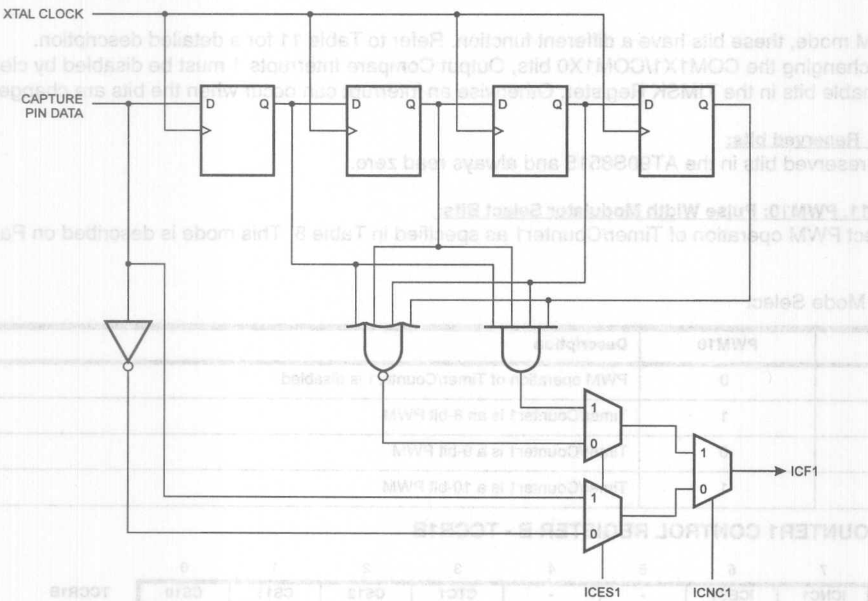


Figure 36. The Input Capture Noise Canceler

If the noise canceler function is enabled, the actual trigger condition for the capture event is monitored over 4 samples before the capture is activated. The input pin signal is sampled at XTAL clock frequency.

THE TIMER/COUNTER1 CONTROL REGISTER A - TCCR1A

Bit	7	6	5	4	3	2	1	0	
$2F ($4F)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7,6 - COM1A1, COM1A0 : Compare Output Mode1A, bits 1 and 0:
The COM1A1 and COM1A0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1A - Output CompareA pin 1. Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) to control an output pin. The control configuration is shown in Table 7.

Bits 5,4 - COM1B1, COM1B0 : Compare Output Mode1B, bits 1 and 0:
The COM1B1 and COM1B0 control bits determine any output pin action following a compare match in Timer/Counter1. Any output pin actions affect pin OC1B - Output CompareB. Since this is an alternative function to an I/O port, the corresponding direction control bit must be set (one) to control an output pin. The following control configuration is given:

Table 7. Compare 1 Mode Select

COM1X1	COM1X0	Description
0	0	Timer/Counter1 disconnected from output pin OC1X
0	1	Toggle the OC1X output line.
1	0	Clear the OC1X output line (to zero).
1	1	Set the OC1X output line (to one).

X = A or B

In PWM mode, these bits have a different function. Refer to Table 11 for a detailed description.
When changing the COM1X1/COM1X0 bits, Output Compare Interrupts 1 must be disabled by clearing their Interrupt Enable bits in the TIMSK Register. Otherwise an interrupt can occur when the bits are changed.

Bits 3..2 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and always read zero.

Bits 1..0 - PWM11, PWM10: Pulse Width Modulator Select Bits:
These bits select PWM operation of Timer/Counter1 as specified in Table 8. This mode is described on Page 5-41.

Table 8. PWM Mode Select

PWM11	PWM10	Description
0	0	PWM operation of Timer/Counter1 is disabled
0	1	Timer/Counter1 is an 8-bit PWM
1	0	Timer/Counter1 is a 9-bit PWM
1	1	Timer/Counter1 is a 10-bit PWM

THE TIMER/COUNTER1 CONTROL REGISTER B - TCCR1B

Bit	7	6	5	4	3	2	1	0	
$2E ($4E)	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R	R/w	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ICNC1 : Input Capture1 Noise Canceler (4 CKs):
When the ICNC1 bit is cleared (zero), the input capture trigger noise canceler function is disabled. The input capture is triggered at the first rising/falling edge sampled on the ICP - input capture pin - as specified. When the ICNC1 bit is set (one), four successive samples are measures on the ICP - input capture pin, and all samples must be high/low according to the input capture trigger specification in the ICES1 bit. The actual sampling frequency is XTAL clock frequency.

Bit 6 - ICES1 : Input Capture1 Edge Select:
While the ICES1 bit is cleared (zero), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the falling edge of the input capture pin - ICP. While the ICES1 bit is set (one), the Timer/Counter1 contents are transferred to the Input Capture Register - ICR1 - on the rising edge of the input capture pin - ICP.

Bits 5, 4 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and always read zero.

Bit 3 - CTC1 : Clear Timer/Counter1 on Compare match:
When the CTC1 control bit is set (one), the Timer/Counter1 is reset to $0000 in the clock cycle after a compareA match. If the CTC1 control bit is cleared, the Timer/Counter1 continues counting until it is stopped, cleared, wraps around (overflow) or changes direction. In PWM mode, this bit has no effect.

Bits 2,1,0 - CS12, CS11, CS10 : Clock Select1, bit 2,1 and 0:
The Clock Select1 bits 2,1 and 0 define the prescaling source of Timer/Counter1.

Table 9. Clock 1 Prescale Select

CS12	CS11	CS10	Description
0	0	0	Stop, the Timer/Counter1 is stopped.
0	0	1	CK
0	1	0	CK / 8
0	1	1	CK / 64
1	0	0	CK / 256
1	0	1	CK / 1024
1	1	0	External Pin T1, falling edge
1	1	1	External Pin T1, rising edge

The Stop condition provides a Timer Enable/Disable function. The CK down divided modes are scaled directly from the CK oscillator clock. If the external pin modes are used, the corresponding setup must be performed in the actual direction control register (cleared to zero gives an input pin).

THE TIMER/COUNTER1 - TCNT1H AND TCNT1L

Bit	15	14	13	12	11	10	9	8	
$2D ($4D)	MSB								TCNT1H
$2C ($4C)								LSB	TCNT1L
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

This 16-bit register contains the prescaled value of the 16-bit Timer/Counter1. To ensure that both the high and low bytes are read and written simultaneously when the CPU accesses these registers, the access is performed using an 8-bit temporary register (TEMP).

- **TCNT1 Timer/Counter1 Write:**

  When the CPU writes to the high byte TCNT1H, the written data is placed in the TEMP register. Next, when the CPU writes the low byte TCNT1L, this byte of data is combined with the byte data in the TEMP register, and all 16 bits are written to the TCNT1 Timer/Counter1 register simultaneously. Consequently, the high byte TCNT1H must be accessed first for a full 16-bit register write operation.

• **TCNT1 Timer/Counter1 Read:**
When the CPU reads the low byte TCNT1L, the data of the low byte TCNT1L is sent to the CPU and the data of the high byte TCNT1H is placed in the TEMP register. When the CPU reads the data in the high byte TCNT1H, the CPU receives the data in the TEMP register. Consequently, the low byte TCNT1L must be accessed first for a full 16-bit register read operation.

The Timer/Counter1 is realized as an up or up/down (in PWM mode) counter with read and write access. If Timer/Counter1 is written to and a clock source is selected, the Timer/Counter1 continues counting in the timer clock cycle after it is preset with the written value.

TIMER/COUNTER1 OUTPUT COMPARE REGISTER - OCR1AH AND OCR1AL

Bit	15	14	13	12	11	10	9	8	
$2B ($4B)	MSB								OCR1AH
$2A ($4A)								LSB	OCR1AL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

TIMER/COUNTER1 OUTPUT COMPARE REGISTER - OCR1BH AND OCR1BL

Bit	15	14	13	12	11	10	9	8	
$29 ($49)	MSB								OCR1BH
$28 ($48)								LSB	OCR1BL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The output compare registers are 16-bit read/write registers.

The Timer/Counter1 Output Compare Registers contain the data to be continuously compared with Timer/Counter1. Actions on compare matches are specified in the Timer/Counter1 Control and Status register.

Since the Output Compare Registers - OCR1A and OCR1B - are 16-bit registers, a temporary register TEMP is used when OCR1A/B are written to ensure that both bytes are updated simultaneously. When the CPU writes the high byte, OCR1AH or OCR1BH, the data is temporarily stored in the TEMP register. When the CPU writes the low byte, OCR1AL or OCR1BL, the TEMP register is simultaneously written to OCR1AH or OCR1BH. Consequently, the high byte OCR1AH or OCR1BH must be written first for a full 16-bit register write operation.

THE TIMER/COUNTER1 INPUT CAPTURE REGISTER - ICR1H AND ICR1L

Bit	15	14	13	12	11	10	9	8	
$25 ($45)	MSB								ICR1H
$24 ($44)								LSB	ICR1L
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

The input capture register is a 16-bit read-only register.

When the rising or falling edge (according to the input capture edge setting - ICES1) of the signal at the input capture pin - ICP - is detected, the current value of the Timer/Counter1 is transferred to the Input Capture Register - ICR1. At the same time, the input capture flag - ICF1 - is set (one).

Since the Input Capture Register - ICR1 - is a 16-bit register, a temporary register TEMP is used when ICR1 is read to ensure that both bytes are read simultaneously. When the CPU reads the low byte ICR1L, the data is sent to the CPU and the data of the high byte ICR1H is placed in the TEMP register. When the CPU reads the data in the high byte ICR1H, the CPU receives the data in the TEMP register. Consequently, the low byte ICR1L must be accessed first for a full 16-bit register read operation.

TIMER/COUNTER1 IN PWM MODE

When the PWM mode is selected, Timer/Counter1 and the Output Compare Register1A - OCR1A and the Output Compare Register1B - OCR1B, form a dual 8, 9 or 10-bit, free-running, glitch-free and phase correct PWM with outputs on the PD5(OC1A) and OC1B pins. Timer/Counter1 acts as an up/down counter, counting up from $0000 to TOP (see Table 10) , when it turns and counts down again to zero before the cycle is repeated. When the counter value matches the contents of the 10 least significant bits of OCR1A or OCR1B, the PD5(OC1A)/OC1B pins are set or cleared according to the settings of the COM1A1/COM1A0 or COM1B1/COM1B0 bits in the Timer/Counter1 Control Register TCCR1A. Refer to Table 11 for details.

Table 10. Timer TOP Values and PWM Frequency

PWM Resolution	Timer TOP value	Frequency
8-bit	$00FF (255)	$f_{TC1}/510$
9-bit	$01FF (511)	$f_{TC1}/1022$
10-bit	$03FF(1023)	$f_{TC1}/2046$

Table 11. Compare1 Mode Select in PWM Mode

COM1X1	COM1X0	Effect on OCX1
0	0	Not connected
0	1	Not connected
1	0	Cleared on compare match, upcounting. Set on compare match, downcounting (non-inverted PWM).
1	1	Cleared on compare match, downcounting. Set on compare match, upcounting (inverted PWM).

Note: X = A or B

Note that in the PWM mode, the 10 least significant OCR1A/OCR1B bits, when written, are transferred to a temporary location. They are latched when Timer/Counter1 reaches the value TOP. This prevents the occurrence of odd-length PWM pulses (glitches) in the event of an unsynchronized OCR1A/OCR1B write. See Figure 37 for an example.

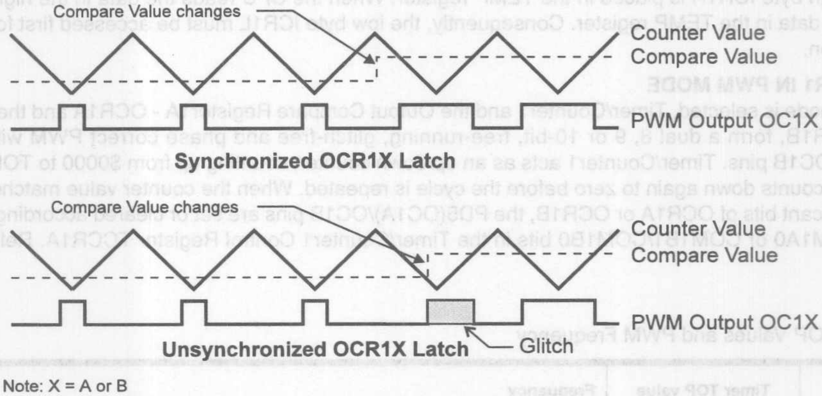


Figure 37. Effects on Unsynchronized OCR1 Latching

When OCR1 contains $0000 or TOP, the output OC1A/OC1B is held low or high according to the settings of COM1A1/COM1A0 or COM1B1/COM1B0. This is shown in Table 12:

Table 12. PWM Outputs OCR1X = $0000 or TOP

COM1X1	COM1X0	OCR1X	Output OC1X
1	0	$0000	L
1	0	TOP	H
1	1	$0000	H
1	1	TOP	L

Note: X = A or B

In PWM mode, the Timer Overflow Flag1, TOV1, is set when the counter changes direction at $0000. Timer Overflow Interrupt1 operates exactly as in normal Timer/Counter mode, i.e. it is executed when TOV1 is set provided that Timer Overflow Interrupt1 and global interrupts are enabled. This does also apply to the Timer Output Compare1 flags and interrupts.

The Watchdog Timer

The Watchdog Timer is clocked from a separate on-chip oscillator which runs at 1MHz This is the typical value at V_{CC} = 5V. See characterization data for typical values at other V_{CC} levels. By controlling the Watchdog Timer prescaler, the Watchdog reset interval can be adjusted from 16 to 2048 ms. The WDR - Watchdog Reset - instruction resets the Watchdog Timer. Eight different clock cycle periods can be selected to determine the reset period. If the reset period expires without another Watchdog reset, the AT90S8515 resets and executes from the reset vector. For timing details on the Watchdog reset, refer to Page 5-28.

To prevent unintentional disabling of the watchdog, a special turn-off secuence must be followed when the watchdog is disabled.Refer to the description of the Watchdog Timer Control Regiter for details

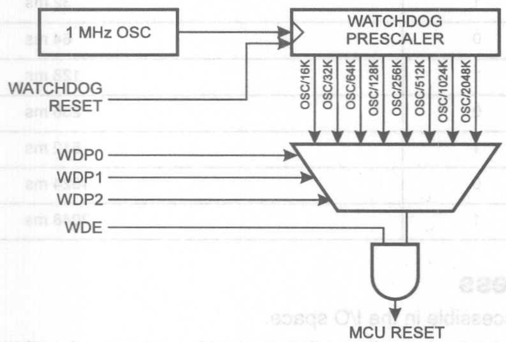


Figure 38. Watchdog Timer

THE WATCHDOG TIMER CONTROL REGISTER - WDTCR

Bit	7	6	5	4	3	2	1	0	
$21 ($41)	-	-	-	WDTTOE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bits 7..5 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and will always read as zero.

Bit 4 - WDTOE : Watch Dog Turn-Off Enable:
This bit must be set (one) when the WDE bit is cleared. Otherwise, the watchdog will not be disabled. Once set, hardware will clear this bit to zero after four clock cycles. Refer to the description of the WDE bit for a watchdog disable procedure.

Bit 3 - WDE : Watch Dog Enable:
When the WDE is set (one) the Watchdog Timer is enabled, and if the WDE is cleared (zero) the Watchdog Timer function is disabled. WDE can only be cleared if the WDTOE bit is set(one). To disable an enabled watchdog timer, the following procedure must be followed:

1. In the same operation, write a logical one to WDTOE and WDE. A logcal one must be written to WDE even though it is set to one before the disable operation starts.
2. Within the next four clock cycles, write a logical 0 to WDE. This disables the watchdog.

<u>Bits 2..0 - WDP2, WDP1, WDP0 : Watch Dog Timer Prescaler 2, 1 and 0:</u>
The WDP2, WDP1 and WDP0 bits determine the Watchdog Timer prescaling when the Watchdog Timer is enabled. The different prescaling values and their corresponding Timeout Periods are shown in Table 13.

Table 13. Watch Dog Timer Prescale Select (Typical Values at V_{CC} = 5V)

WDP2	WDP1	WDP0	Timeout Period
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

EEPROM Read/Write Access

The EEPROM access registers are accessible in the I/O space.

The write access time is in the range of 2.5 - 4ms, depending on the V_{CC} voltages. A self-timing function, however, lets the user software detect when the next byte can be written. An EEPROM brown-out detection prevents writing to the EEPROM if V_{CC} is below a certain level.

In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.

When the EEPROM is read or written, the CPU is halted for two clock cycles before the next instruction is executed.

THE EEPROM ADDRESS REGISTER - EEARH AND EEARL

Bit	15	14	13	12	11	10	9	8	
$1F ($3F)	-	-	-	-	-	-	-	EEAR9	EEARH
$1E ($3E)	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	EEARL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	8	0	0	0	0	0	0	0	
	8	0	0	0	0	0	0	0	

The EEPROM Address Registers - EEARH and EEARL specify the EEPROM address in the 512 bytes EEPROM space. The EEPROM data bytes are addressed linearly between 0 and 512.

THE EEPROM DATA REGISTER - EEDR

Bit	7	6	5	4	3	2	1	0	
$1D ($3D)	MSB							LSB	EEDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

<u>Bits 7..0 - EEDR7..0 : EEPROM Data:</u>
For the EEPROM write operation, the EEDR register contains the data to be written to the EEPROM in the address given by the EEAR register. For the EEPROM read operation, the EEDR contains the data read out from the EEPROM at the address given by EEAR.

THE EEPROM CONTROL REGISTER - EECR

Bit	7	6	5	4	3	2	1	0	
$1C ($3C)	-	-	-	-	-	EEMWE	EEWE	EERE	EECR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7..3 - Res : Reserved bits:
These bits are reserved bits in the AT90S2313 and will always read as zero.

Bit 2 - EEMWE : EEPROM Master Write Enable:
The EEMWE bit determines whether setting EEWE to one causes the EEPROM to be written. When EEMWE is set(one) setting EEWE will write data to the EEPROM at the selected address If EEMWE is zero, setting EEWE will have no effect. When EEMWE has been set (one) by software, hardware clears the bit to zero after four clock cycles. See the description of the EEWE bit for a EEPROM write procedure.

Bit 1 - EEWE : EEPROM Write Enable:
The EEPROM Write Enable Signal EEWE is the write strobe to the EEPROM. When address and data are correctly set up, the EEWE bit must be set to write the value into the EEPROM. The EEMWE bit must be set when the logical one is written to EEWE, otherwise no EEPROM write takes place. The following procedure should be followed when writing the EEPROM (the order of steps 2 and 3 is unessential):

1. Wait until EEWE becomes zero.
2. Write new EEPROM address to EEARL and EEARH (optional)
3. Write new EEPROM data to EEDR (optional)
4. Write a logical one to the EEMWE bit in EECR
5. Within four clock cycles after setting EEMWE, write a logical one to EEWE.

When the write access time (typically 2.5 ms at V_{CC} = 5V or 4 ms at V_{CC} = 2.7V) has elapsed, the EEWE bit is cleared (zero) by hardware. The user software can poll this bit and wait for a zero before writing the next byte. When EEWE has been set, the CPU is halted for two cycles before the next instruction is executed.

Bit 0 - EERE : EEPROM Read Enable:
The EEPROM Read Enable Signal EERE is the read strobe to the EEPROM. When the correct address is set up in the EEAR register, the EERE bit must be set. When the EERE bit is cleared (zero) by hardware, requested data is found in the EEDR register. The EEPROM read access takes one instruction and there is no need to poll the EERE bit. When EERE has been set, the CPU is halted for two cycles before the next instruction is executed.

The user should poll the EEWE bit before starting the read operation. If a write operation is in progress when new data or address is written to the EEPROM I/O registers, the write operation will be interrupted, and the result is undefined.

The Serial Peripheral Interface - SPI

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT90S8515 and peripheral devices or between several AT90S8515 devices. The AT90S8515 SPI features include the following:

- Full-Duplex, 3-Wire Synchronous Data Transfer
- Master or Slave Operation
- 5 Mbit/s Bit Frequency (max.)
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wakeup from Idle Mode (Slave Mode Only)

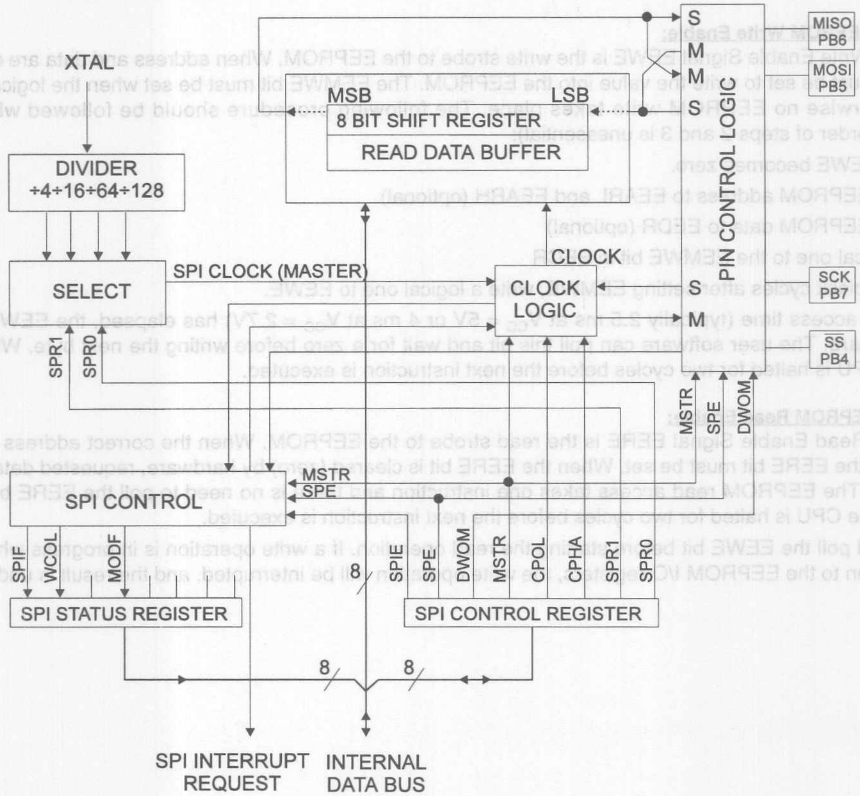


Figure 39. SPI Block Diagram

The interconnection between master and slave CPUs with SPI is shown in Figure 40. The PB7(SCK) pin is the clock output in the master mode and is the clock input in the slave mode. Writing to the SPI data register of the master CPU starts the SPI clock generator, and the data written shifts out of the PB5(MOSI) pin and into the PB5(MOSI) pin of the slave CPU. After shifting one byte, the SPI clock generator stops, setting the end of transmission flag (SPIF). If the SPI interrupt enable bit (SPIE) in the SPCR register becomesset, an interrupt is requested. The Slave Select input, PB4(SS), is set low to select an individual SPI device as a slave. The two shift registers in the Master and the Slave can be considered as one distributed 16-bit circular shift register. This is shown in Figure 40. When data is shifted from the master to the slave, data is also shifted in the opposite direction, simultaneously. This means that during one shift cycle, data in the master and the slave are interchanged.

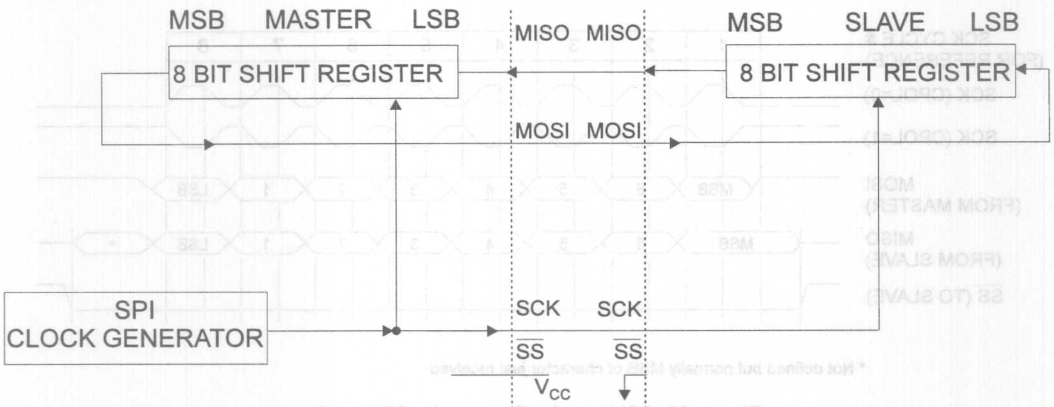


Figure 40. SPI Master-Slave Interconnection

The system is single buffered in the transmit direction and double buffered in the receive direction. This means that characters to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first character is lost.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK and $\overline{SS}$ pins is overriden according to the following table:

Table 14.SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

$\overline{SS}$ Pin Functionality

When the SPI is configured as a master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin. If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. If $\overline{SS}$ is configured as an input, it must be hold high to ensure Master SPI operatin. If, in master mode, the $\overline{SS}$ pin is input, and is driven low by peripheral circuitry, the SPI system interpretes this as that another master selects the SPI as a slave and will start sending data to it. To avoid bus contention, the SPI system takes the following actions:

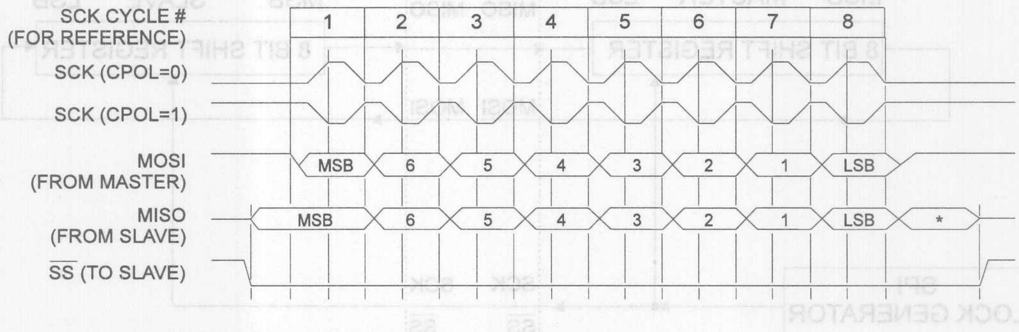
1. The MSTR bit in SPCR is cleared and the SPI system becomes a slave. As a result of the SPI becoming a slave, the MOSI and SCK pins become inputs.
2. The SPIF flag in SPSR is set, and if the SPI interrupt is enabled, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmittal is used in master mode, and there exists a possibility that SS is driven low, the interrupt should always check that the MSTR bit is still set. Once the MSTR bit has been cleared by a slave select, it must be set by the user.

When the SPI is configured as a slave, the SS is always input. When SS is held low, the SPI is activated and MISO becomes an output if configured so by the user. All other pins are inputs When SS is driven low, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data.

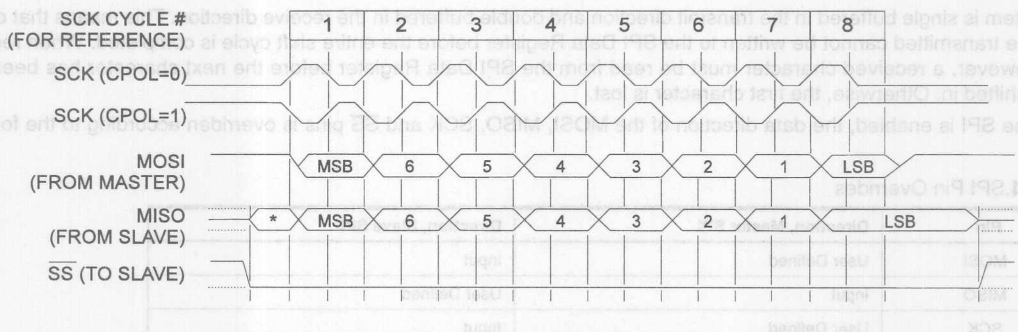
Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. The SPI data transfer formats are shown in Figure 41 and Figure 42.



* Not defined but normally MSB of character just received

Figure 41. SPI Transfer Format with CPHA = 0



* Not defined but normally LSB of previously transmitted character

Figure 42. SPI Transfer Format with CPHA = 1

THE SPI CONTROL REGISTER - SPCR

Bit	7	6	5	4	3	2	1	0	
$0D ($2D)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	1	0	0	

Bit 7 - SPIE : SPI Interrupt Enable:
This bit causes setting of the SPIF bit in the SPSR register to execute the SPI interrupt provided that global interrupts are enabled.

Bit 6 - SPE : SPI Enable:
When the SPE bit is set (one), the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 - DORD : Data ORDer:
When the DORD bit is set (one), the LSB of the data word is transmitted first.

When the DORD bit is cleared (zero), the MSB of the data word is transmitted first.

Bit 4 - MSTR : Master/Slave Select:
This bit selects Master SPI mode when set (one), and Slave SPI mode when cleared (zero). If $\overline{SS}$ is configured as an input and is driven low whil MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI master mode.

Bit 3 - CPOL : Clock POLarity:
When this bit is set (one), SCK is high when idle. When CPOL is cleared (zero), SCK is low when idle. Refer to Figure 41 and Figure 42 for additional information.

Bit 2 - CPHA : Clock PHAse:
Refer to Figure 41 or Figure 42 for the functionality of this bit.

Bits 1,0 - SPR1, SPR0 : SPI Clock Rate Select 1 and 0:
These two bits control the SCK rate of the device configured as a master. SPR1 and SPR2 have no effect on the slave. The relationship between SCK and the Oscillator Clock frequency f_{cl} is shown in the following table:

Table 15. Relationship Between SCK and the Oscillator Frequency

SPR1	SPR0	SCK Frequency
0	0	f_{cl} / 4
0	1	f_{cl} / 16
1	0	f_{cl} / 64
1	1	f_{cl} / 128

THE SPI STATUS REGISTER - SPSR

Bit	7	6	5	4	3	2	1	0	
$0E ($2E)	SPIF	WCOL	-	-	-	-	-	-	SPSR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - SPIF : SPI Interrupt Flag:
When a serial transfer is complete, the SPIF bit is set (one) and an interrupt is generated if SPIE in SPCR is set (one) and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in master mode, this will also set the SPIF

flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI status register with SPIF set (one), then accessing the SPI Data Register (SPDR).

Bit 6 - WCOL : Write COLlision flag:
The WCOL bit is set if the SPI data register (SPDR) is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it will have no effect. The WCOL bit (and the SPIF bit) are cleared (zero) by first reading the SPI Status Register with WCOL set (one), and then accessing the SPI Data Register.

Bit 5..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and will always read as zero.

The SPI interface on the AT90S8515 is also used for program memory and EEPROM downloading or uploading. See Page 5-78 for serial programming and verification.

THE SPI DATA REGISTER - SPDR

Bit	7	6	5	4	3	2	1	0	
$0F ($2F)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The SPI Data Register is a read/write register used for data transfer between the register file and the SPI Shift register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.

The UART

The AT90S8515 features a full duplex Universal Asynchronous Receiver and Transmitter (UART). The main features are:

- Baud rate generator generates any baud rate
- High baud rates at low XTAL frequencies
- 8 or 9 bits data
- Noise filtering
- Overrun detection
- Framing Error detection
- False Start Bit detection
- Three separate interrupts on TX Complete, TX Data Register Empty and RX Complete

Data Transmission

A block schematic of the UART transmitter is shown in Figure 43.

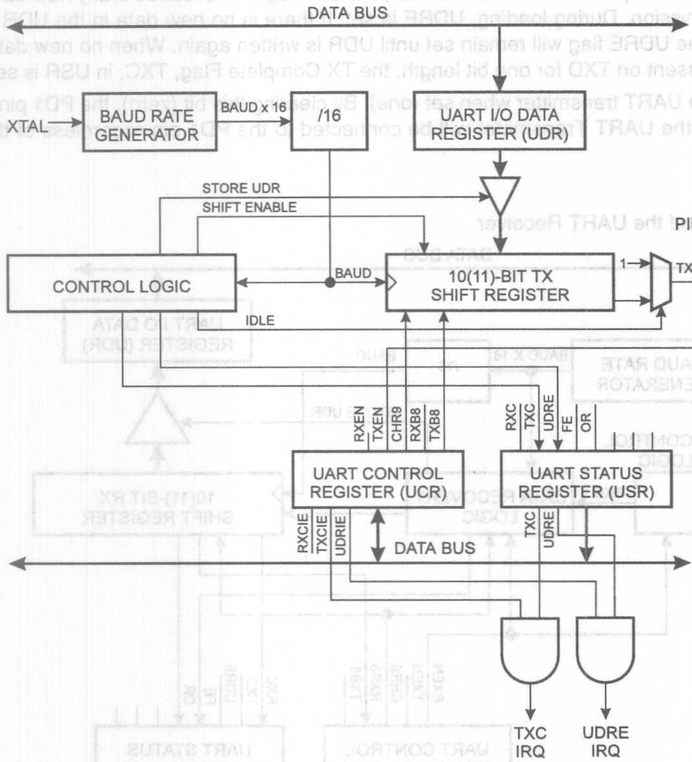


Figure 43. UART Transmitter

Data transmission is initiated by writing the data to be transmitted to the UART I/O Data Register, UDR. Data is transferred from UDR to the Transmit shift register when:

- A new character has been written to UDR after the stop bit from the previous character has been shifted out. The shift register is loaded immediately.
- A new character has been written to UDR before the stop bit from the previous character has been shifted out. The shift register is loaded when the stop bit of the character currently being transmitted has been shifted out.

If the 10(11)-bit Transmitter shift register is empty or when, data is transferred from UDR to the shift register. At this time the UDRE (UART Data Register Empty) bit in the UART Status Register, USR, is set. When this bit is set (one), the UART is ready to receive the next character. At the same time as the data is transferred from UDR to the 10(11)-bit shift register, bit 0 of the shift register is cleared (start bit) and bit 9 or 10 is set (stop bit). If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the TXB8 bit in UCR is transferred to bit 9 in the Transmit shift register.

On the Baud Rate clock following the transfer operation to the shift register, the start bit is shifted out on the TXD pin. Then follows the data, LSB first. When the stop bit has been shifted out, the shift register is loaded if any new data has been written to the UDR during the transmission. During loading, UDRE is set. If there is no new data in the UDR register to send when the stop bit is shifted out, the UDRE flag will remain set until UDR is written again. When no new data has been written, and the stop bit has been present on TXD for one bit length, the TX Complete Flag, TXC, in USR is set.

The TXEN bit in UCR enables the UART transmitter when set (one). By clearing this bit (zero), the PD1 pin can be used for general I/O. When TXEN is set, the UART Transmitter will be connected to the PD1 pin regardless of the setting of the DDD1 bit in DDRB.

Data Reception

Figure 44 shows a block diagram of the UART Receiver

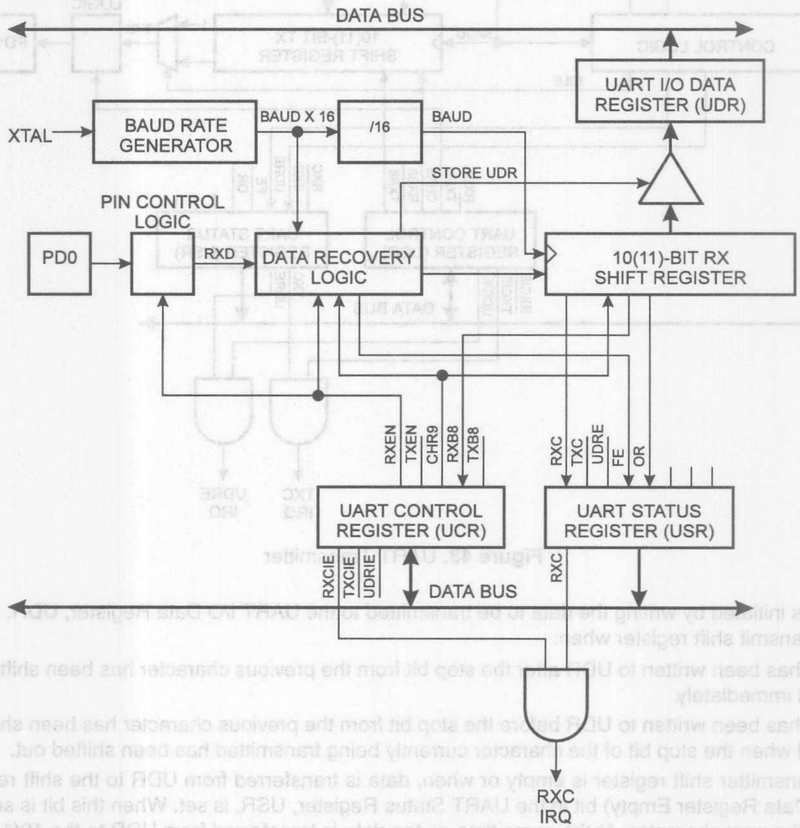


Figure 44. UART Receiver

The receiver front-end logic samples the signal on the RXD pin at a frequency 16 times the baud rate. While the line is idle, one single sample of logical zero will be interpreted as the falling edge of a start bit, and the start bit detection sequence is initiated. Let sample 1 denote the first zero-sample. Following the 1 to 0-transition, the receiver samples the RXD pin at samples 8, 9 and 10. If two or more of these three samples are found to be logical ones, the start bit is rejected as a noise spike and the receiver starts looking for the next 1 to 0-transition.

If however, a valid start bit is detected, sampling of the data bits following the start bit is performed. These bits are also sampled at samples 8, 9 and 10. The logical value found in at least two of the three samples is taken as the bit value. All bits are shifted into the transmitter shift register as they are sampled. Sampling of an incoming character is shown in Figure 45.

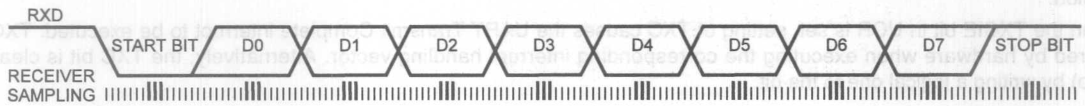


Figure 45. Sampling Received Data

When the stop bit enters the receiver, the majority of the three samples must be one to accept the stop bit. If two or more samples are logical zeros, the Framing Error (FE) flag in the UART Status Register (USR) is set. Before reading the UDR register, the user should always check the FE bit to detect Framing Errors.

Whether or not a valid stop bit is detected at the end of a character reception cycle, the data is transferred to UDR and the RXC flag in USR is set. UDR is in fact two physically separate registers, one for transmitted data and one for received data. When UDR is read, the Receive Data register is accessed, and when UDR is written, the Transmit Data register is accessed. If 9 bit data word is selected (the CHR9 bit in the UART Control Register, UCR is set), the RXB8 bit in UCR is loaded with bit 9 in the Transmit shift register when data is transferred to UDR.

If, after having received a character, the UDR register has not been read since the last receive, the OverRun (OR) flag in UCR is set. This means that the last data byte shifted into to the shift register could not be transferred to UDR and has been lost. The OR bit is buffered, and is updated when the valid data byte in UDR is read. Thus, the user should always check the OR bit after reading the UDR register in order to detect any overruns.

By clearing the RXEN bit in the UCR register, the receiver is disabled. This means that the PD0 pin can be used as a general I/O pin. When RXEN is set, the UART Receiver will be connected to the PD0 pin regardless of the setting of the DDD0 bit in DDRB.

UART Control

THE UART I/O DATA REGISTER - UDR

Bit	7	6	5	4	3	2	1	0	
$0C ($2C)	MSB							LSB	UDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UDR register is actually two physically separate registers sharing the same I/O address. When writing to the register, the UART Transmit Data register is written. When reading from UDR, the UART Receive Data register is read.

THE UART STATUS REGISTER - USR

Bit	7	6	5	4	3	2	1	0	
$0B ($2B)	RXC	TXC	UDRE	FE	OR	-	-	-	USR
Read/Write	R	R	R	R	R	R	R	R	
Initial value	0	0	1	0	0	0	0	0	

The USR register is a read-only register providing information on the UART Status.

Bit 7 - RXC: UART Receive Complete:

This bit is set (one) when a received character is transferred from the Receiver Shift register to UDR. The bit is set regardless of any detected framing errors. When the RXCIE bit in UCR is set, the UART Receive Complete interrupt will be executed when RXC is set(one). RXC is cleared by reading UDR. When interrupt-driven data reception is used, the UART Receive Complete Interrupt routine must read UDR in order to clear RXC, otherwise a new interrupt will occur once the interrupt routine terminates.

Bit 6 - TXC : UART Transmit Complete:
This bit is set (one) when the entire character (including the stop bit) in the Transmit Shift register has been shifted out and no new data has been written to UDR. This flag is especially useful in half-duplex communications interfaces, where a transmitting application must enter receive mode and free the communications bus immediately after completing the transmission.

When the TXCIE bit in UCR is set, setting of TXC causes the UART Transmit Complete interrupt to be executed. TXC is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the TXC bit is cleared (zero) by writing a logical one to the bit.

Bit 5 - UDRE : UART Data Register Empty:
This bit is set (one) when a character written to UDR is transferred to the Transmit shift register. Setting of this bit indicates that the transmitter is ready to receive a new character for transmission.

When the UDRIE bit in UCR is set, the UART Transmit Complete interrupt to be executed as long as UDRE is set. UDRE is cleared by writing UDR. When interrupt-driven data transmittal is used, the UART Data Register Empty Interrupt routine must write UDR in order to clear UDRE, otherwise a new interrupt will occur once the interrupt routine terminates.

UDRE is set (one) during reset to indicate that the transmitter is ready.

Bit 4 - FE : Framing Error:
This bit is set if a Framing Error condition is detected, i.e. when the stop bit of an incoming character is zero.

The FE bit is cleared when the stop bit of received data is one.

Bit 3 - OR : OverRun:
This bit is set if an Overrun condition is detected, i.e. when a character already present in the UDR register is not read before the next character has been shifted into the Receiver Shift register. The OR bit is buffered, which means that it will be set once the valid data still in UDRE is read.

The OR bit is cleared (zero) when data is received and transferred to UDR.

Bits 2..0 - Res : Reserved bits:
These bits are reserved bits in the AT90S8515 and will always read as zero.

THE UART CONTROL REGISTER - UCR

Bit	7	6	5	4	3	2	1	0	
$0A ($2A)	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	UCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - RXCIE : RX Complete Interrupt Enable:
When this bit is set (one), a setting of the RXC bit in USR will cause the Receive Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 6 - TXCIE : TX Complete Interrupt Enable:
When this bit is set (one), a setting of the TXC bit in USR will cause the Transmit Complete interrupt routine to be executed provided that global interrupts are enabled.

Bit 5 - UDRIE : UART Data Register Empty Interrupt Enable:
When this bit is set (one), a setting of the UDRE bit in USR will cause the UART Data Register Empty interrupt routine to be executed provided that global interrupts are enabled.

Bit 4 - RXEN : Receiver Enable:
This bit enables the UART receiver when set (one). When the receiver is disabled, the TXC, OR and FE status flags cannot become set. If these flags are set, turning off RXEN does not cause them to be cleared.

Bit 3 - TXEN : Transmitter Enable:
This bit enables the UART transmitter when set (one). When disabling the transmitter while transmitting a character, the transmitter is not disabled before the character in the shift register plus any following character in UDR has been completely transmitted.

Bit 2 - CHR9 : 9 Bit Characters:
When this bit is set (one) transmitted and received characters are 9 bit long plus start and stop bits. The 9th bit is read and written by using the RXB8 and TXB8 bits in UCR, respectively. The 9th data bit can be used as an extra stop bit or a parity bit.

Bit 1 - RXB8 : Receive Data Bit 8
When CHR9 is set (one), RXB8 is the 9th data bit of the received character.

Bit 0 - TXB8 : Transmit Data Bit 8
When CHR9 is set (one), TXB8 is the 9th data bit in the character to be transmitted.

THE BAUD RATE GENERATOR

The baud rate generator is a frequency divider which generates baud-rates according to the following equation:

$$\mathrm{BAUD} = \frac{f_{CK}}{16(\mathrm{UBRR}+1)}$$

- BAUD = Baud-Rate
- fck= Crystal Clock frequency
- UBRR= Contents of the UART Baud Rate register, UBRR (0-255)

For standard crystal frequencies, the most commonly used baud rates can be generated by using the UBRR settings in Table 16. UBRR values which yield an actual baud rate differing less than 2% from the target baud rate, are bolded in the table.

Table 16. UBRR Settings at Various Crystal Frequencies

Baud Rate	1 MHz		%Error	1.8432 MHz		%Error	2 MHz		%Error	2.4576 MHz		%Error
2400	UBRR=	25	0.2	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	63	0.0
4800	UBRR=	12	0.2	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	31	0.0
9600	UBRR=	6	7.5	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	15	0.0
14400	UBRR=	3	7.8	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	10	3.1
19200	UBRR=	2	7.8	UBRR=	5	0.0	UBRR=	6	7.5	UBRR=	7	0.0
28800	UBRR=	1	7.8	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	6.3
57600	UBRR=	0	7.8	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	12.5
115200	UBRR=	0	84.3	UBRR=	0	0.0	UBRR=	0	7.8	UBRR=	0	25.0

Baud Rate	3.2768 MHz		%Error	3.6864 MHz		%Error	4 MHz		%Error	4.608 MHz		%Error
2400	UBRR=	84	0.4	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0
4800	UBRR=	42	0.8	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0
9600	UBRR=	20	1.6	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0
14400	UBRR=	13	1.6	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0
19200	UBRR=	10	3.1	UBRR=	11	0.0	UBRR=	12	0.2	UBRR=	14	0.0
28800	UBRR=	6	1.6	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0
57600	UBRR=	3	12.5	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0
115200	UBRR=	1	12.5	UBRR=	1	0.0	UBRR=	1	7.8	UBRR=	2	20.0

Baud Rate	7.3728 MHz		%Error	8 MHz		%Error	9.216 MHz		%Error	11.059 MHz		%Error
2400	UBRR=	191	0.0	UBRR=	207	0.2	UBRR=	239	0.0	UBRR=	287	-
4800	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0	UBRR=	143	0.0
9600	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0	UBRR=	71	0.0
14400	UBRR=	31	0.0	UBRR=	34	0.8	UBRR=	39	0.0	UBRR=	47	0.0
19200	UBRR=	23	0.0	UBRR=	25	0.2	UBRR=	29	0.0	UBRR=	35	0.0
28800	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0	UBRR=	23	0.0
57600	UBRR=	7	0.0	UBRR=	8	3.7	UBRR=	9	0.0	UBRR=	11	0.0
115200	UBRR=	3	0.0	UBRR=	3	7.8	UBRR=	4	0.0	UBRR=	5	0.0

Baud Rate	14.746 MHz		%Error	16 MHz		%Error	18.432 MHz		%Error	20 MHz		%Error
2400	UBRR=	383	-	UBRR=	416	-	UBRR=	479	-	UBRR=	520	-
4800	UBRR=	191	0.0	UBRR=	207	0.2	UBRR=	239	0.0	UBRR=	259	-
9600	UBRR=	95	0.0	UBRR=	103	0.2	UBRR=	119	0.0	UBRR=	129	0.2
14400	UBRR=	63	0.0	UBRR=	68	0.6	UBRR=	79	0.0	UBRR=	86	0.2
19200	UBRR=	47	0.0	UBRR=	51	0.2	UBRR=	59	0.0	UBRR=	64	0.2
28800	UBRR=	31	0.0	UBRR=	34	0.8	UBRR=	39	0.0	UBRR=	42	0.9
57600	UBRR=	15	0.0	UBRR=	16	2.1	UBRR=	19	0.0	UBRR=	21	1.4

THE UART BAUD RATE REGISTER - UBRR

Bit	7	6	5	4	3	2	1	0	
\$09 (\$29)	MSB							LSB	UBRR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

The UBRR register is an 8-bit read/write register which specifies the UART Baud Rate according to the equation on the previous page.

The Analog Comparator

The analog comparator compares the input values on the positive pin PB2 (AIN0) and negative pin PB3 (AIN1). When the voltage on the positive pin PB2 (AIN0) is higher than the voltage on the negative pin PB3 (AIN1), the Analog Comparator Output, ACO is set (one). The comparator's output can be set to trigger the Timer/Counter1 Input Capture function. In addition, the comparator can trigger a separate interrupt, exclusive to the Analog Comparator. The user can select Interrupt triggering on comparator output rise, fall or toggle. A block diagram of the comparator and its surrounding logic is shown in Figure 46.

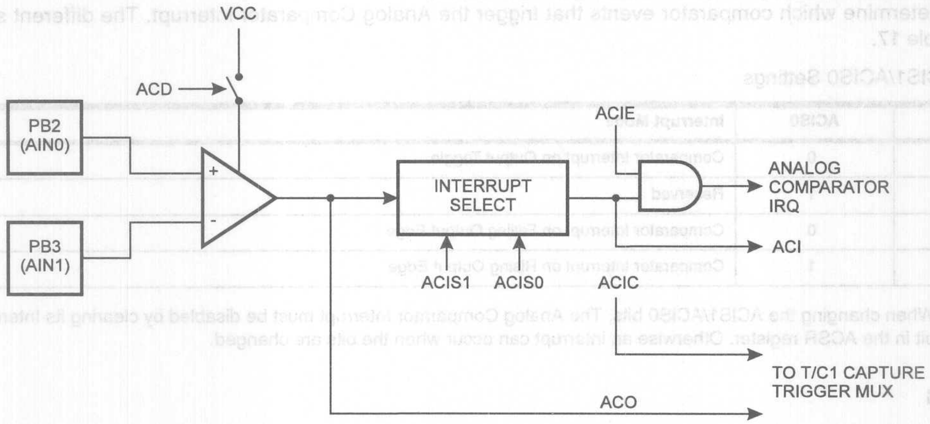


Figure 46. Analog Comparator Block Diagram

THE ANALOG COMPARATOR CONTROL AND STATUS REGISTER - ACSR

Bit	7	6	5	4	3	2	1	0	
$08 ($28)	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	ACSR
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

Bit 7 - ACD : Analog Comparator Disable
When this bit is set(one), the power to the analog comparator is switched off. This bit can be set at any time to turn off the analog comparator. It is most commonly used if power consumption during Idle Mode is critical, and wake-up from the analog comparator is not required. When changing the ACD bit, the Analog Comparator Interrupt must be disabled by clearing the ACIE bit in ACSR. Otherwise an interrupt can occur when the bit is changed.

Bit 6 - Res : Reserved bit:
This bit is a reserved bit in the AT90S8515 and will always read as zero.

Bit 5 - ACO : Analog Comparator Output:
ACO is directly connected to the comparator output.

Bit 4 - ACI : Analog Comparator Interrupt Flag:
This bit is set (one) when a comparator output event triggers the interrupt mode defined by ACI1 and ACI0. The Analog Comparator Interrupt routine is executed if the ACIE bit is set (one) and the I-bit in SREG is set (one). ACI is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ACI is cleared by writing a logic one to the flag.

Bit 3 - ACIE : Analog Comparator Interrupt Enable:
When the ACIE bit is set (one) and the I-bit in the Status Register is set (one), the analog comparator interrupt is activated. When cleared (zero), the interrupt is disabled.

Bit 2 - ACIC : Analog Comparator Input Capture enable:
When set (one), this bit enables the Input Capture function in Timer/Counter1 to be triggered by the analog comparator. The comparator output is in this case directly connected to the Input Capture front-end logic, making the comparator utilize the noise canceler and edge select features of the Timer/Counter1 Input Capture interrupt. When cleared (zero), no connection between the analog comparator and the Input Capture function is given. To make the comparator trigger the Timer/Counter1 Input Capture interrupt, the TICIE1 bit in the Timer Interrupt Mask Register (TIMSK) must be set (one).

Bits 1,0 - ACIS1, ACIS0 : Analog Comparator Interrupt Mode Select:
These bits determine which comparator events that trigger the Analog Comparator interrupt. The different settings are shown in Table 17.

Table 17. ACIS1/ACIS0 Settings

ACIS1	ACIS0	Interrupt Mode
0	0	Comparator Interrupt on Output Toggle
0	1	Reserved
1	0	Comparator Interrupt on Falling Output Edge
1	1	Comparator Interrupt on Rising Output Edge

Note: When changing the ACIS1/ACIS0 bits, The Analog Comparator Interrupt must be disabled by clearing its Interrupt Enable bit in the ACSR register. Otherwise an interrupt can occur when the bits are changed.

I/O-Ports

Port A

PORT A is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port A, one each for the Data Register - PORTA, $1B($3B), Data Direction Register - DDRA, $1A($3A) and the Port A Input Pins - PINA, $19($39). The Port A Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The PORT A output buffers can sink 20mA and thus drive LED displays directly. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The PORT A pins have alternate functions related to the optional external data SRAM. PORT A can be configured to be the multiplexed low-order address/data bus during accesses to the external data memory. In this mode, PORT A has internal pullups.

When PORT A is set to the alternate function by the SRE - External SRAM Enable - bit in the MCUCR - MCU Control Register, the alternate settings override the data direction register.

THE PORT A DATA REGISTER - PORTA

Bit	7	6	5	4	3	2	1	0	
\$1B (\$3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT A DATA DIRECTION REGISTER - DDRA

Bit	7	6	5	4	3	2	1	0	
\$1A (\$3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT A INPUT PINS ADDRESS - PINA

Bit	7	6	5	4	3	2	1	0	
\$19 (\$39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port A Input Pins address - PINA - is not a register, and this address enables access to the physical value on each Port A pin. When reading PORTA the PORTA Data Latch is read, and when reading PINA, the logical values present on the pins are read.

PORTA AS GENERAL DIGITAL I/O

All 8 bits in PORT A are equal when used as digital I/O pins.

PAn, General I/O pin: The DDAn bit in the DDRA register selects the direction of this pin, if DDAn is set (one), PAn is configured as an output pin. If DDAn is cleared (zero), PAn is configured as an input pin. If PORTAn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTAn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 18. DDAn Effects on PORT A Pins

DDAn	PORTAn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PAn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

PORT A SCHEMATICS

Note that all port pins are synchronized. The synchronization latch is however, not shown in the figure.

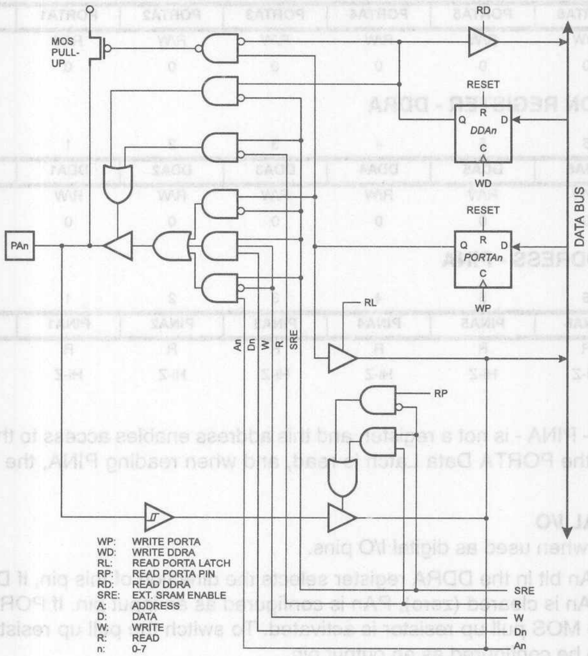


Figure 47. PORTA Schematic Diagrams (Pins PA0 - PA7)

Port B

Port B is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port B, one each for the Data Register - PORTB, $18($38), Data Direction Register - DDRB, $17($37) and the Port B Input Pins - PINB, $16($36). The Port B Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The Port B output buffers can sink 20mA and thus drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The Port B pins with alternate functions are shown in the following table:

Table 19. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB0	T0 (Timer/Counter 0 external counter input)
PB1	T1 (Timer/Counter 1 external counter input)
PB2	AIN0 (Analog comparator positive input)
PB3	AIN1 (Analog comparator negative input)
PB4	$\overline{SS}$ (SPI Slave Select input)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB7	SCK (SPI Bus Serial Clock)

When the pins are used for the alternate function the DDRB and PORTB register has to be set according to the alternate function description.

THE PORT B DATA REGISTER - PORTB

Bit	7	6	5	4	3	2	1	0	
$18 ($38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B DATA DIRECTION REGISTER - DDRB

Bit	7	6	5	4	3	2	1	0	
$17 ($37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT B INPUT PINS ADDRESS - PINB

Bit	7	6	5	4	3	2	1	0	
$16 ($36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port B Input Pins address - PINB - is not a register, and this address enables access to the physical value on each Port B pin. When reading PORTB, the PORTB Data Latch is read, and when reading PINB, the logical values present on the pins are read.

PORTB AS GENERAL DIGITAL I/O

All 8 bits in port B are equal when used as digital I/O pins.

PBn, General I/O pin: The DDBn bit in the DDRB register selects the direction of this pin, if DDBn is set (one), PBn is configured as an output pin. If DDBn is cleared (zero), PBn is configured as an input pin. If PORTBn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, the PORTBn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 20. DDBn Effects on Port B Pins

DDBn	PORTBn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PBn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

ALTERNATE FUNCTIONS OF PORTB

The alternate pin configuration is as follows:

SCK - PORTB, Bit 7:
SCK: Master clock output, slave clock input pin for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB7. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB7. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB7 bit. See the description of the SPI port for further detatils.

MISO - PORTB, Bit 6:
MISO: Master data input, slave data output pin for SPI channel. When the SPI is enabled as a master, this pin is configured as an input regardless of the setting of DDB6. When the SPI is enabled as a slave, the data direction of this pin is controlled by DDB6. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB6 bit. See the description of the SPI port for further detatils.

MOSI - PORTB, Bit 5:
MOSI: SPI Master data output, slave data input for SPI channel. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further detatils.

SS - PORTB, Bit 4:
$\overline{SS}$: Slave port select input. When the SPI is enabled as a slave, this pin is configured as an input regardless of the setting of DDB5. As a slave, the SPI is activated when this pin is driven low. When the SPI is enabled as a master, the data direction of this pin is controlled by DDB5. When the pin is forced to be an input, the pull-up can still be controlled by the PORTB5 bit. See the description of the SPI port for further detatils.

AIN1 - PORTB, Bit 3
AIN1, Analog Comparator Negative Input. When configured as an input (DDB3 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB3 is cleared (zero)), this pin also serves as the negative input of the on-chip analog comparator.

AIN0 - PORTB, Bit 2
AIN0, Analog Comparator Positive Input. When configured as an input (DDB2 is cleared (zero)) and with the internal MOS pull up resistor switched off (PB2 is cleared (zero)), this pin also serves as the positive input of the on-chip analog comparator.

T1 - PORTB, Bit 1:
T1, Timer/Counter1 counter source. See the timer description for further details

T0 - PORTB, Bit 0:
T0: Timer/Counter0 counter source. See the timer description for further details.

PORT B SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

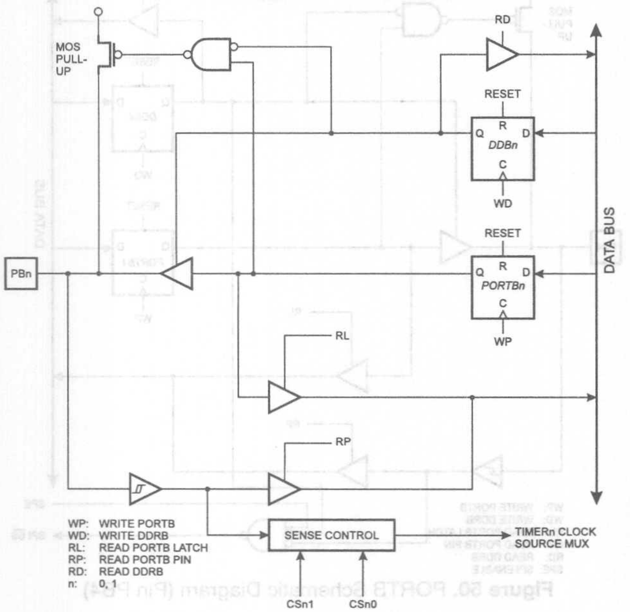


Figure 48. PORTB Schematic Diagram (Pins PB0 and PB1)

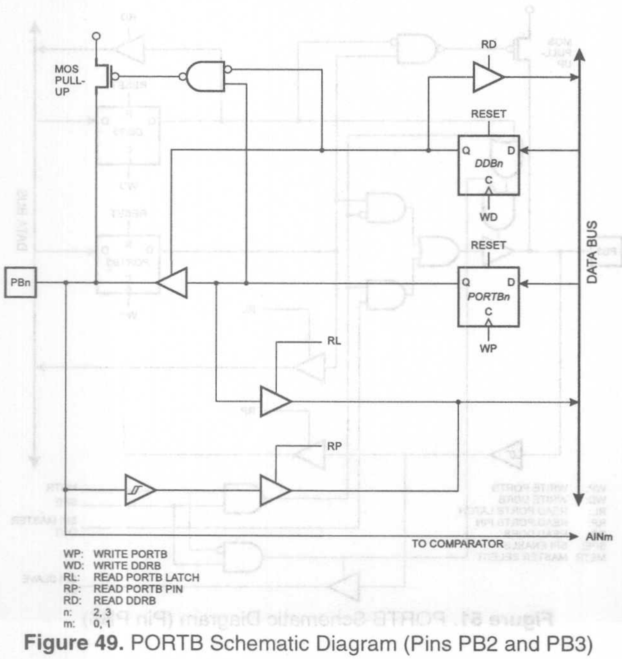


Figure 49. PORTB Schematic Diagram (Pins PB2 and PB3)

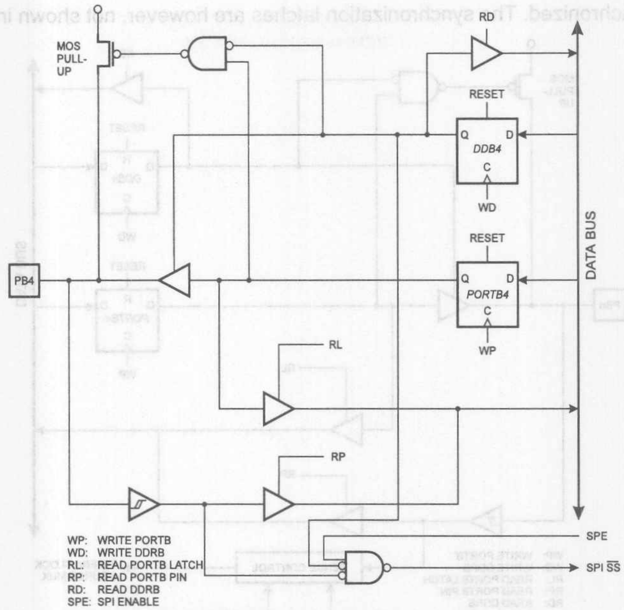


Figure 50. PORTB Schematic Diagram (Pin PB4)

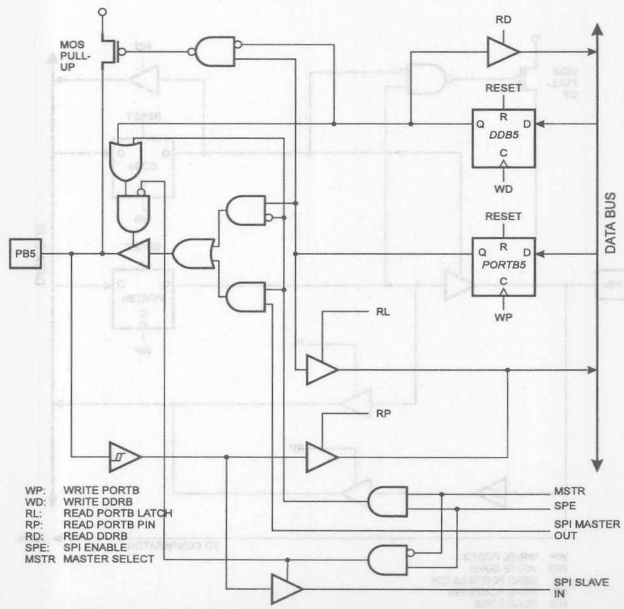


Figure 51. PORTB Schematic Diagram (Pin PB5)

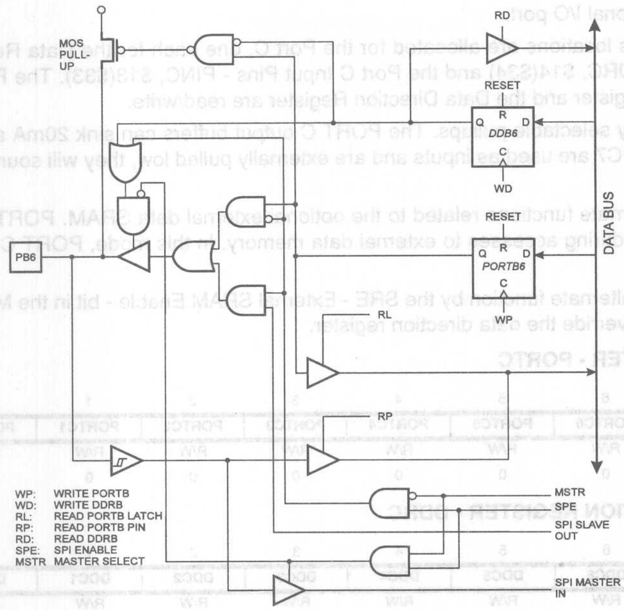


Figure 52. PORTB Schematic Diagram (Pin PB6)

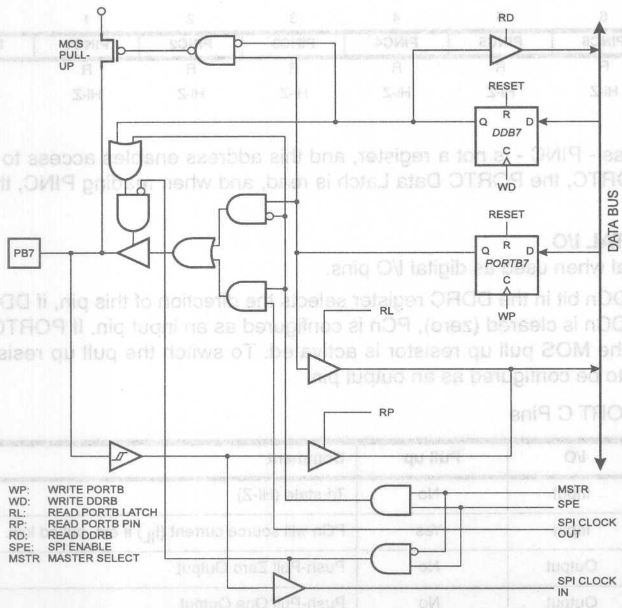


Figure 53. PORTB Schematic Diagram (Pin PB7)

5

Port C

PORT C is an 8-bit bi-directional I/O port.

Three data memory address locations are allocated for the Port C, one each for the Data Register - PORTC, $15($35), Data Direction Register - DDRC, $14($34) and the Port C Input Pins - PINC, $13($33). The Port C Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

All port pins have individually selectable pullups. The PORT C output buffers can sink 20mA and thus drive LED displays directly. When pins PC0 to PC7 are used as inputs and are externally pulled low, they will source current (I_{IL}) if the internal pullups are activated.

The PORT C pins have alternate functions related to the optional external data SRAM. PORT C can be configured to be the high-order address byte during accesses to external data memory. In this mode, PORT C uses internal pullups when emitting 1's.

When PORT C is set to the alternate function by the SRE - External SRAM Enable - bit in the MCUCR - MCU Control Register, the alternate settings override the data direction register.

THE PORT C DATA REGISTER - PORTC

Bit	7	6	5	4	3	2	1	0	
$15 ($35)	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT C DATA DIRECTION REGISTER - DDRC

Bit	7	6	5	4	3	2	1	0	
$14 ($34)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT C INPUT PINS ADDRESS - PINC

Bit	7	6	5	4	3	2	1	0	
$13 ($33)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port C Input Pins address - PINC - is not a register, and this address enables access to the physical value on each Port C pin. When reading PORTC, the PORTC Data Latch is read, and when reading PINC, the logical values present on the pins are read.

PORTC AS GENERAL DIGITAL I/O

All 8 bits in PORT C are equal when used as digital I/O pins.

PCn, General I/O pin: The DDCn bit in the DDRC register selects the direction of this pin, if DDCn is set (one), PCn is configured as an output pin. If DDCn is cleared (zero), PCn is configured as an input pin. If PORTCn is set (one) when the pin configured as an input pin, the MOS pull up resistor is activated. To switch the pull up resistor off, PORTCn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 21. DDCn Effects on PORT C Pins

DDCn	PORTCn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PCn will source current (I_{IL}) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7…0, pin number

PORT C SCHEMATICS

Note that all port pins are synchronized. The synchronization latch is however, not shown in the figure.

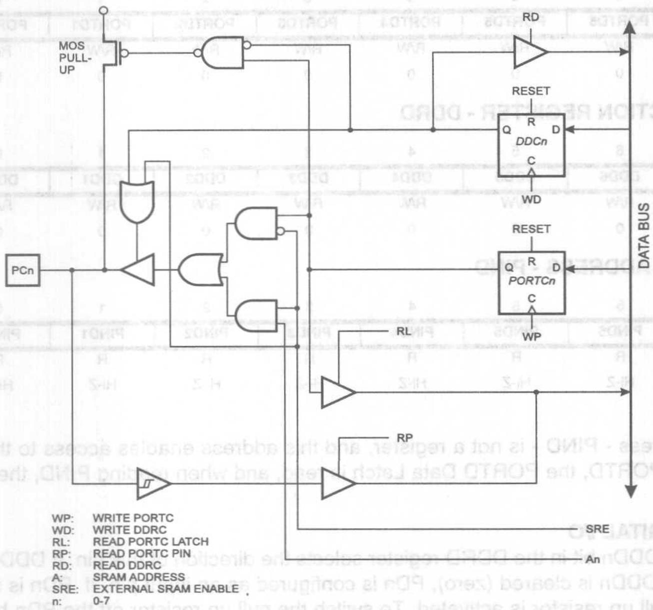


Figure 54. PORTC Schematic Diagram (Pins PC0 - PC7)

Port D

Port D is an 8 bit bi-directional I/O port with internal pullups.

Three data memory address locations are allocated for the Port D, one each for the Data Register - PORTD, $12($32), Data Direction Register - DDRD, $11($31) and the Port D Input Pins - PIND, $10($30). The Port D Input Pins address is read only, while the Data Register and the Data Direction Register are read/write.

The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current (I_{IL}) if the pullups are activated.

Some Port D pins have alternate functions as shown in the following table:

Table 22. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD0	RDX (UART Input line)
PD1	TDX (UART Output line)
PD2	INT0 (External interrupt 0 input)
PD3	INT1 (External interrupt 1 input)
PD5	OC1A (Timer/Counter1 Output compareA match output)
PD6	$\overline{WR}$ (Write strobe to external memory)
PD7	$\overline{RD}$ (Read strobe to external memory)

When the pins are used for the alternate function the DDRD and PORTD register has to be set according to the alternate function description.

THE PORT D DATA REGISTER - PORTD

Bit	7	6	5	4	3	2	1	0	
$12 ($32)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D DATA DIRECTION REGISTER - DDRD

Bit	7	6	5	4	3	2	1	0	
$11 ($31)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial value	0	0	0	0	0	0	0	0	

THE PORT D INPUT PINS ADDRESS - PIND

Bit	7	6	5	4	3	2	1	0	
$10 ($30)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial value	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z	

The Port D Input Pins address - PIND - is not a register, and this address enables access to the physical value on each Port D pin. When reading PORTD, the PORTD Data Latch is read, and when reading PIND, the logical values present on the pins are read.

PORTD AS GENERAL DIGITAL I/O

PDn, General I/O pin: The DDDn bit in the DDRD register selects the direction of this pin. If DDDn is set (one), PDn is configured as an output pin. If DDDn is cleared (zero), PDn is configured as an input pin. If PDn is set (one) when configured as an input pin the MOS pull up resistor is activated. To switch the pull up resistor off the PDn has to be cleared (zero) or the pin has to be configured as an output pin.

Table 23. DDDn Bits on Port D Pins

DDDn	PORTDn	I/O	Pull up	Comment
0	0	Input	No	Tri-state (Hi-Z)
0	1	Input	Yes	PDn will source current (IIL) if ext. pulled low.
1	0	Output	No	Push-Pull Zero Output
1	1	Output	No	Push-Pull One Output

n: 7,6…0, pin number.

ALTERNATE FUNCTIONS OF PORTD

$\overline{\text{RD}}$ - PORTD, Bit 7:
$\overline{\text{RD}}$ is the external data memory read control strobe.

$\overline{\text{WR}}$ - PORTD, Bit 6:
$\overline{\text{WR}}$ is the external data memory write control strobe.

OC1- PORTD, Bit 5:
OC1, Output compare match output: The PD5 pin can serve as an external output when the Timer/Counter1 compare matches. The PD5 pin has to be configured as an output (DDD5 set (one)) to serve this function. See the Timer/Counter1 description for further details, and how to enable the output. The OC1 pin is also the output pin for the PWM mode timer function.

INT1 - PORTD, Bit 3:

INT1, External Interrupt source 1: The PD3 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

INT0 - PORTD, Bit 2:

INT0, External Interrupt source 0: The PD2 pin can serve as an external interrupt source to the MCU. See the interrupt description for further details, and how to enable the source.

TXD - PORTD, Bit 1:

Transmit Data (Data output pin for the UART). When the UART transmitter is enabled, this pin is configured as an output regardless of the value of DDRD1.

RXD - PORTD, Bit 0:

Receive Data (Data input pin for the UART). When the UART receiver is enabled this pin is configured as an output regardless of the value of DDRD0. When the UART forces this pin to be an input, a logical one in PORTD0 vill turn on the internal pull-up.

PORTD SCHEMATICS

Note that all port pins are synchronized. The synchronization latches are however, not shown in the figures.

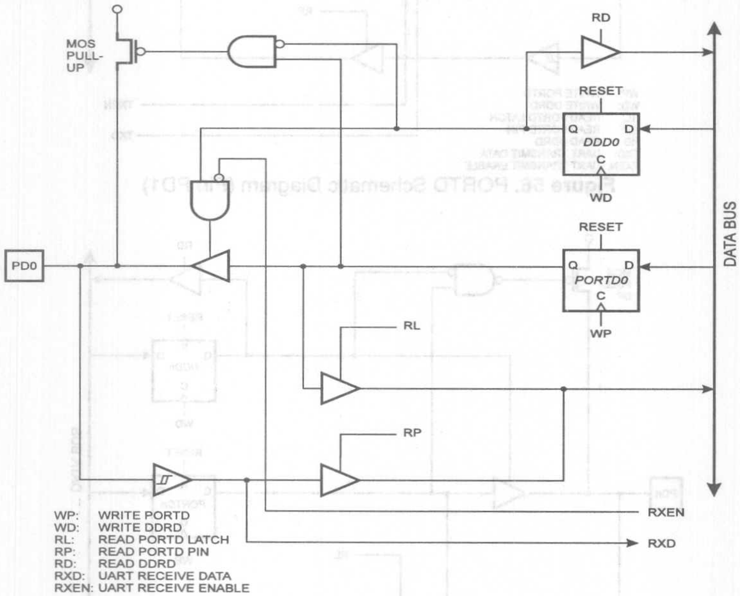


Figure 55. PORTD Schematic Diagram (Pin PD0)

5

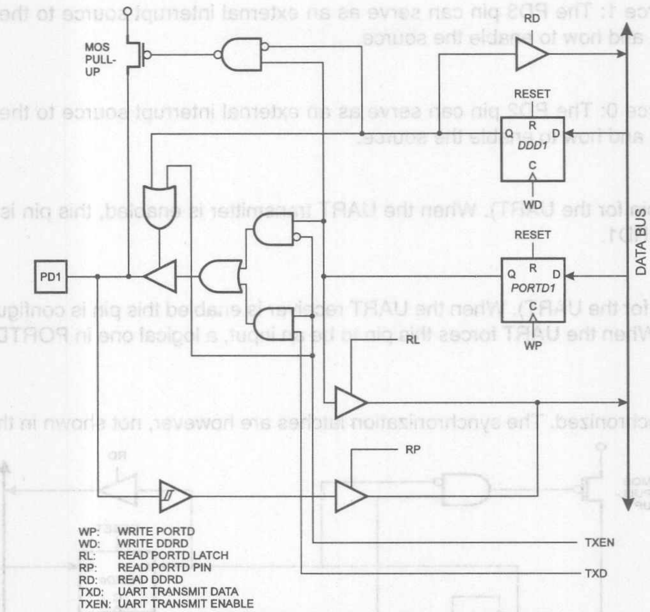


Figure 56. PORTD Schematic Diagram (Pin PD1)

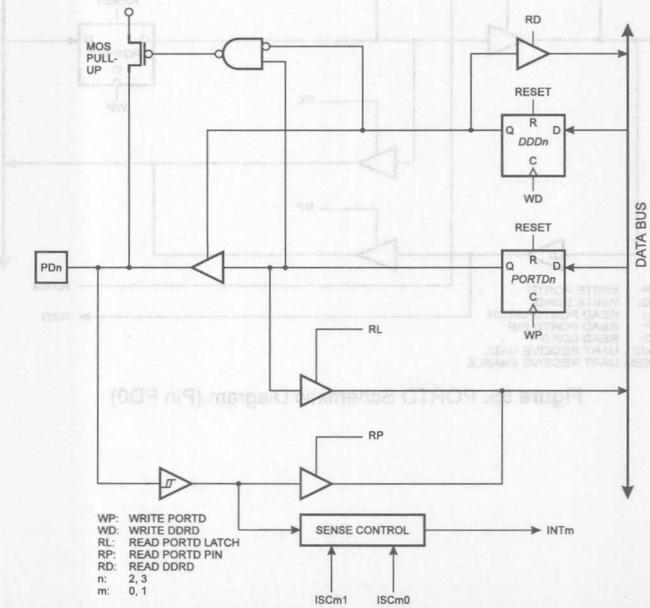


Figure 57. PORTD Schematic Diagram (Pins PD2 and PD3)

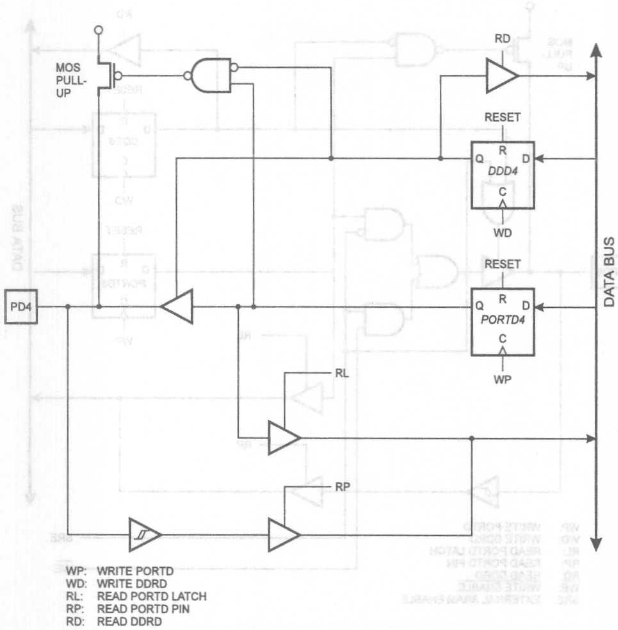


Figure 58. PORTD Schematic Diagram (Pin PD4)

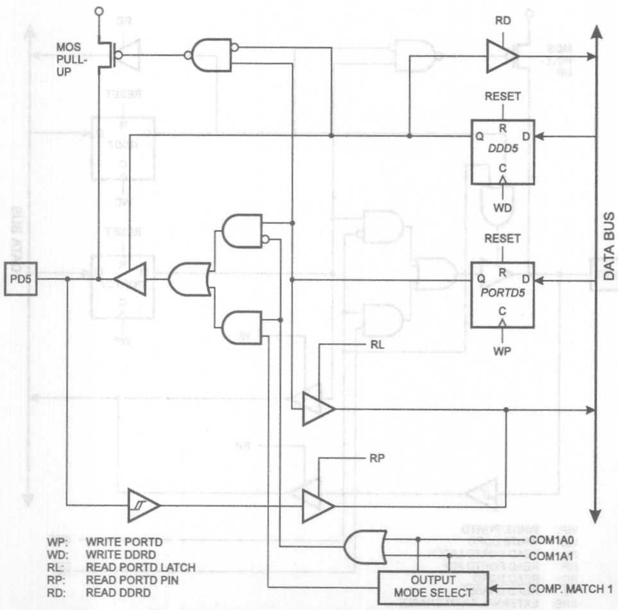


Figure 59. PORTD Schematic Diagram (Pin PD5)

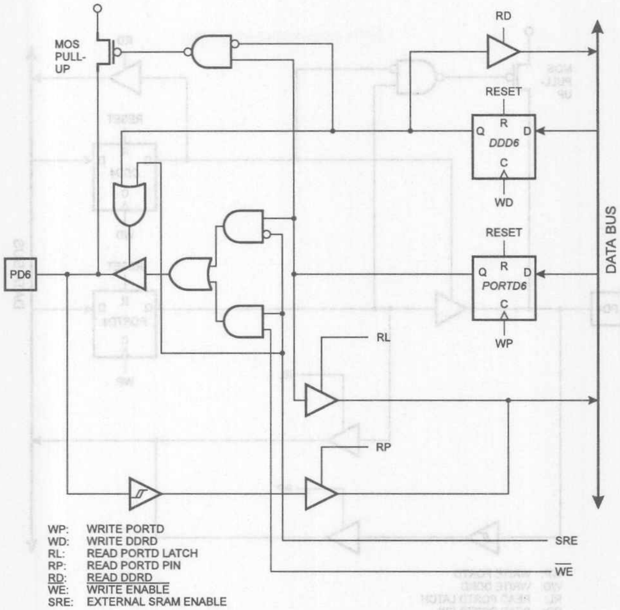


Figure 60. PORTD Schematic Diagram (Pin PD6)

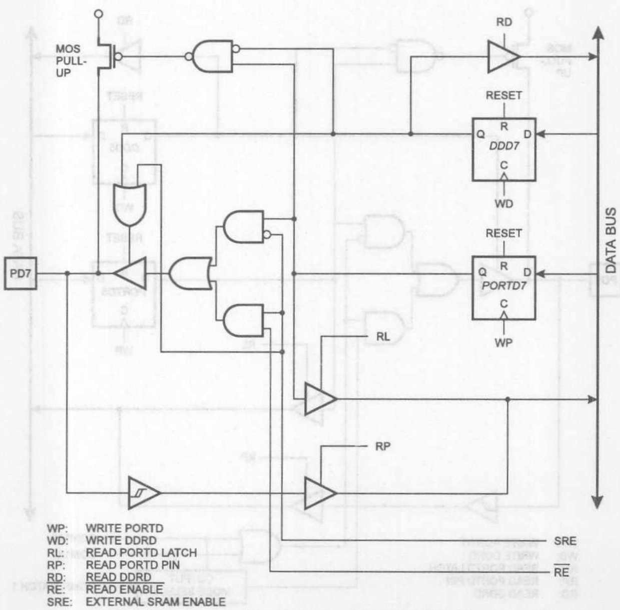


Figure 61. PORTD Schematic Diagram (Pin PD7)

Memory Programming

Program Memory Lock Bits

The AT90S8515 MCU provides two lock bits which can be left unprogrammed ('1') or can be programmed ('0') to obtain the additional features listed in Table 24.

Table 24. Lock Bit Protection Modes

Program Lock Bits			Protection Type
Mode	LB1	LB2	
1	1	1	No program lock features
2	0	1	Further programming of the Flash is disabled
3	0	0	Same as mode 2, but verify is also disabled.

Note: The Lock Bits can only be erased with the Chip Erase operation.

Fuse Bits

The AT90S8515 has two fuse bits, SPIEN and FSTRT.

- When SPIEN is programmed ('0'), Serial Program Downloading is enabled. Default value is programmed ('0').
- When FSTRT is programmed ('0'), the short start-up time is selected. Default value is unprogrammed ('1'). Parts with this bit pre-programmed ('0') can be delivered on demand.

These bits are not accessible in Serial Programming Mode and are not affected by a chip erase.

Signature Bytes

All Atmel microcontrollers have a three-byte signature code which identifies the device. This code can be read in both serial and parallel mode. The three bytes reside in a separate address space, and for the AT90S8515 they are:

1. \$000: \$1E (indicates manufactured by Atmel)
2. \$001: \$93 (indicates 8kB Flash memory)
3. \$002: \$01 (indicates 90S8515 device when \$001 is \$93)

Programming the Flash and EEPROM

Atmel's AT90S8515 offers 8K bytes of in-system reprogrammable Flash Program memory and 512 bytes of EEPROM Data memory.

The AT90S8515 is normally shipped with the on-chip Flash Program and EEPROM Data memory arrays in the erased state (i.e. contents = \$FF) and ready to be programmed. This device supports a High-Voltage (12V) Parallel programming mode and a Low-Voltage Serial programming mode. The +12V is used for programming enable only, and no current of significance is drawn by this pin. The serial programming mode provides a convenient way to download the Program and Data into the AT90S8515 inside the user's system.

The Program and Data memory arrays on the AT90S8515 are programmed byte-by-byte in either programming modes. For the EEPROM, an auto-erase cycle is provided with the self-timed programming operation in the serial programming mode.

Parallel Programming

This section describes how to parallel program and verify Flash Program memory, EEPROM Data memory + Program Memory Lock bits and Fuse bits in the AT90S8515.

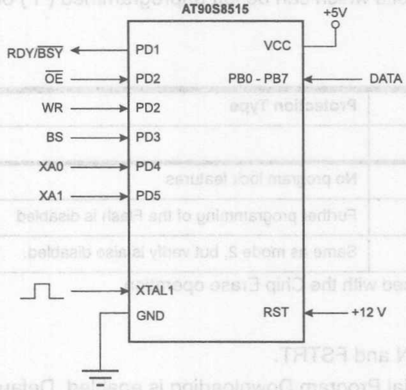


Figure 62. Parallel Programming

SIGNAL NAMES

In this section, some pins of the AT908515 are referenced by signal names describing their functionality during parallel programming rather than their pin names. Pins not described in the following table are referenced by pin names.

Table 25. Pin Name Mapping

Signal Name in Programming Mode	Pin Name	I/O	Function
RDY / $\overline{BSY}$	PD1	O	0: Device is busy programming, 1: Device is ready for new command
$\overline{OE}$	PD2	I	Output Enable (Active Low)
$\overline{WR}$	PD3	I	Write Pulse (Active Low)
BS	PD4	I	Byte Select
XA0	PD5	I	XTAL Action Bit 0
XA1	PD6	I	XTAL Action Bit 1

The XA1/XA0 bits determine the action taken when the XTAL1 pin is given a positive pulse. The bit settings are shown in the following table:

Table 26. XA1 and XA0 Coding

XA1	XA0	Action when XTAL1 is Pulsed
0	0	Load Flash or EEPROM Address (High or Low address byte for Flash determined by BS)
0	1	Load Data (High or Low data byte for Flash determined by BS)
1	0	Load Command
1	1	No Action, Idle

When pulsing $\overline{WR}$ or $\overline{OE}$, the command loaded determines the action on input or output. The command is a byte where the different bits are assigned functions as shown in the following table:

Table 27. Command Byte Bit Coding

Bit#	Meaning when Set
7	Chip Erase
6	Write Fuse Bits. Located in the data byte at the following bit positions: D5: SPIEN Fuse, D0: FSTRT Fuse (Note: Write '0' to program, '1' to erase)
5	Write Lock Bits. Located in the data byte at the following bit positions: D1: LB1, D0: LB2 (Note: write '0' to program)
4	Write Flash or EEPROM (determined by bit 0)
3	Read signature row
2	Read Lock and Fuse Bits. Located in the data byte at the following bits positions: D7: LB1, D6: LB2, D5: SPIEN Fuse, D0: FSTRT Fuse (Note: '0' means programmed)
1	Read from Flash or EEPROM (determined by bit 0)
0	0 : Flash Access, 1 : EEPROM Access

ENTER PROGRAMMING MODE

The following algorithm puts the device in parallel programming mode:

1. Apply 4.5 - 5.5 V between VCC and GND.
2. Set $\overline{RESET}$ and BS pins to '0' and wait at least 100 ns.
3. Apply 12V to and wait at least 100 ns before changing BS.

CHIP ERASE

The chip erase will erase the Flash and EEPROM memories plus Lock bits. The lock bits are not reset until the program memory has been completely erased. The Fuse bits are not changed. A chip erase must be performed before the chip is programmed.

Load Command "Chip Erase"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'.
3. Set PB(7:0) to '1000 0000'. This is the command for Chip erase.
4. Give XTAL1 a positive pulse. This loads the command, and starts the erase of the Flash and EEPROM arrays. After pulsing XTAL1, give $\overline{WR}$ a negative pulse to enable lock bit erase at the end of the erase cycle, then wait for at least 10 ms. Chip erase does not generate any activity on the RDY/$\overline{BSY}$ pin.

PROGRAMMING THE FLASH

Load Command "Program Flash"

1. Set XA1, XA0 to '10'. This enables command loading.
2. Set BS to '0'
3. Set PB(7:0) to '0001 0000'. This is the command for Flash programming.
4. Give XTAL1 a positive pulse. This loads the command.

Load Address Low byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '0'. This selects Low address.
3. Set PB(7:0) = Address Low byte ($00 - $FF)
4. Give XTAL1 a positive pulse. This loads the Address Low byte.

Load Address High byte

1. Set XA1, XA0 to '00'. This enables address loading.
2. Set BS to '1'. This selects High address.
3. Set PB(7:0) = Address High byte ($00 - $0F)
4. Give XTAL1 a positive pulse. This loads the Address High byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data Low byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data Low byte

1. Set BS to ('0').
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY/$\overline{BSY}$ goes low.
3. Wait until RDY/$\overline{BSY}$ goes high to program the next byte.

Load Data byte

1. Set XA1, XA0 to '01'. This enables data loading.
2. Set PB(7:0) = Data High byte ($00 - $FF)
3. Give XTAL1 a positive pulse. This loads the Data byte.

Write Data High byte

1. Set BS to '1'.
2. Give $\overline{WR}$ a negative pulse. This starts programming of the data byte. RDY / $\overline{BSY}$ goes low.
3. Wait until RDY / $\overline{BSY}$ goes high to program the next byte.

The loaded command and address are retained in the device during programming. To simplify programming, the following should be considered.

- The command for Flash programming needs only be loaded before programming of the first byte.
- Address High byte needs only be loaded before programming a new 256 word page in the Flash.

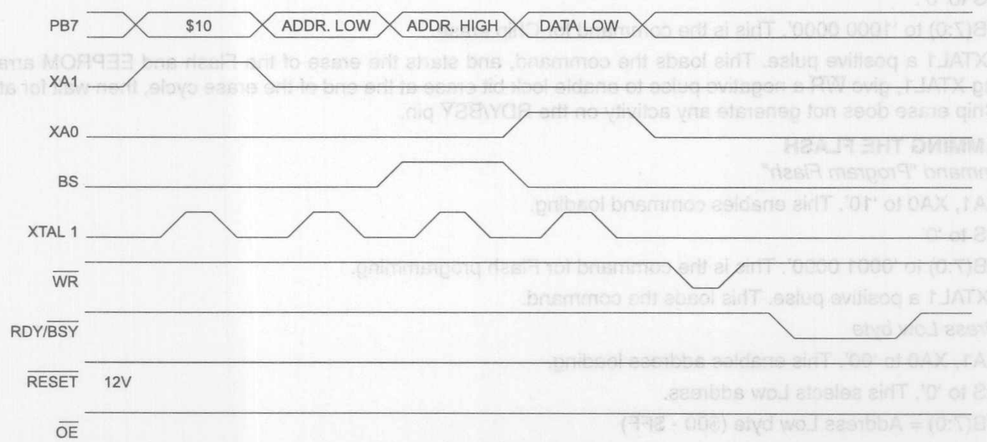


Figure 63. Programming Flash Low Byte

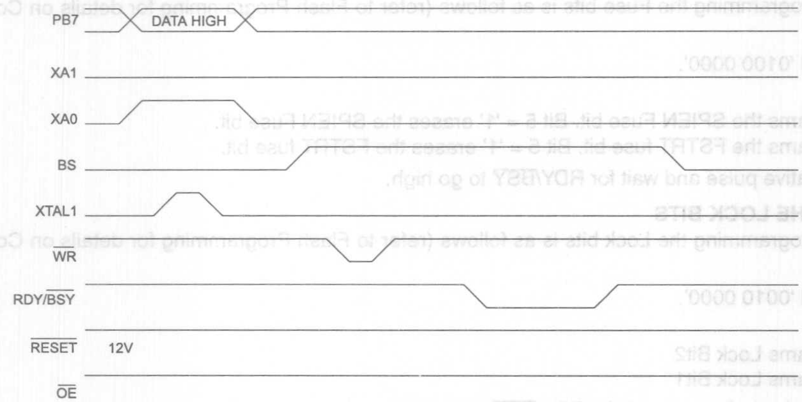


Figure 64. Programming Flash High Byte

PROGRAMMING THE EEPROM

The programming algorithm for the EEPROM data memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0001 0001'.
2. Load Low EEPROM Address ($00 - $FF)
3. Load High EEPROM Address ($00 - $01)
4. Load Low EEPROM Data ($00 - $FF)
5. Give $\overline{\text{WR}}$ a negative pulse and wait for RDY/$\overline{\text{BSY}}$ to go high.

The Command needs only be loaded before programming the first byte.

READING THE FLASH

The algorithm for reading the Flash memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0010'.
2. Load Low Address ($00 - $FF)
3. Load High Address ($00 - $0F)
4. Set $\overline{\text{OE}}$ to '0', and BS to '0'. The Low Data byte can now be read at PB(7:0)
5. Set BS to '1'. The High Data byte can now be read from PB(7:0)
6. Set $\overline{\text{OE}}$ to '1'.

The Command needs only be loaded before reading the first byte.

READING THE EEPROM

The algorithm for reading the EEPROM memory is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0011'.
2. Load Low EEPROM Address ($00 - $FF)
3. Load High EEPROM Address ($00 - $FF)
4. Set $\overline{\text{OE}}$ to '0', and BS to '0'. The EEPROM Data byte can now be read at PB(7:0)
5. Set $\overline{\text{OE}}$ to '1'.

The Command needs only be loaded before reading the first byte.

5

PROGRAMMING THE FUSE BITS

The algorithm for programming the Fuse bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0100 0000'.
2. Load Data.
   Bit 5 = '0' programs the SPIEN Fuse bit. Bit 5 = '1' erases the SPIEN Fuse bit.
   Bit 0 = '0' programs the FSTRT fuse bit. Bit 5 = '1' erases the FSTRT fuse bit.
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

PROGRAMMING THE LOCK BITS

The algorithm for programming the Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0010 0000'.
2. Load Data.
   Bit 2 = '0' programs Lock Bit2
   Bit 1 = '0' programs Lock Bit1
3. Give $\overline{WR}$ a negative pulse and wait for RDY/$\overline{BSY}$ to go high.

The lock bits can only be cleared by executing a chip erase.

READING THE FUSE AND LOCK BITS

The algorithm for reading the Fuse and Lock bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 0100'.
2. Set $\overline{OE}$ to '0', and BS to '1'. The Status of Fuse and Lock bits can now be read at PB(7:0)
   Bit 7: Lock Bit1 ('0' means programmed)
   Bit 6: Lock Bit2 ('0' means programmed)
   Bit 5: SPIEN Fuse ('0' means programmed, '1' means erased)
   Bit 0: FSTRT Fuse ('0' means programmed, '1' means erased)
3. Set $\overline{OE}$ to '1'.

Observe especially that BS needs to be set to '1'.

READING THE SIGNATURE BYTES

The algorithm for reading the Signature Bytes bits is as follows (refer to Flash Programming for details on Command, Address and Data loading):

1. Load Command '0000 1000'.
2. Load Low address ($00 - $02)
3. Set $\overline{OE}$ to '0', and BS to '0'. The Selected Signature byte can now be read at PB(7:0)
4. Set $\overline{OE}$ to '1'.

The command needs only be programmed before reading the first byte.

Serial Downloading

Both the Program and Data memory arrays can be programmed using the serial SPI bus while $\overline{RESET}$ is pulled to GND. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After $\overline{RESET}$ is set low, the Programming Enable instruction needs to be executed first before program/erase operations can be executed.

When programming the EEPROM, an auto-erase cycle is built into the self-timed programming operation (in the serial mode ONLY) and there is no need to first execute the Chip Erase instruction. The Chip Erase operation turns the content of every memory location in both the Program and EEPROM arrays into $FF.

The Program and EEPROM memory arrays have separate address spaces:

$0000 to $0FFF for Program memory and $0000 to $001FF for EEPROM memory.

Either an external system clock is supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2.The minimum low and high periods for the serial clock (SCK) input are defined as follows:

Low: > 1 XTAL1 clock cycle

High: > 4 XTAL1 clock cycles

SERIAL PROGRAMMING ALGORITHM

To program and verify the AT90S8515 in the serial programming mode, the following sequence is recommended (See four byte instruction formats in Table 28):

1. *Power-up sequence:*

   Apply power between VCC and GND while $\overline{\text{RESET}}$ and SCK are set to '0'. (If the programmer can not guarantee that SCK is held low during power-up, $\overline{\text{RESET}}$ must be given a positive pulse after SCK has been set to '0'.) If a crystal is not connected across pins XTAL1 and XTAL2, apply a 0 to 20 MHz clock to the XTAL1 pin.
2. Wait for at least 20 ms and enable serial programming by sending the Programming Enable serial instruction to pin MOSI/PB5. Refer to the above section for minimum high and low periods for the serial clock input (SCK).
3. If a chip erase is performed (must be done to erase the Flash), wait 10ms, give $\overline{\text{RESET}}$ a positive pulse and start over again from Step 2.
4. The Flash or EEPROM array is programmed one byte at a time by supplying the address and data together with the appropriate Write instruction. An EEPROM memory location is first automatically erased before new data is written. The next byte can be written after 4 ms.
5. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/PB6.
6. At the end of the programming session, $\overline{\text{RESET}}$ can be set high to commence normal operation.
7. Power-off sequence (if needed):

   Set XTAL1 to '0' (if a crystal is not used).

   Set $\overline{\text{RESET}}$ to '1'.

   Turn V_{CC} power off

Table 28. Serial Programming Instruction Set

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx	Enable Serial Programming after RESET goes low.
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip erase both 8K & 512byte memory arrays
Read Program Memory	0010 **H**000	xxxx **aaaa**	**bbbb bbbb**	**oooo oooo**	Read **H**(high or low) data **o** from Program memory at word address **a:b**
Write Program Memory	0100 **H**000	xxxx **aaaa**	**bbbb bbbb**	**iiii iiii**	Write **H**(high or low) data **i** to Program memory at word address **a:b**
Read EEPROM Memory	1010 0000	xxxx xxx0	**bbbb bbbb**	**oooo oooo**	Read data **o** from EEPROM memory at address **a:b**
Write EEPROM Memory	1100 0000	xxxx xxx0	**bbbb bbbb**	**iiii iiii**	Write data **i** to EEPROM memory at address **a:b**
Write Lock Bits	1010 1100	111x x*21*x	xxxx xxxx	xxxx xxxx	Write lock bits. Set bits *1,2*='0' to program lock bits.
Read Device Code	0011 0000	xxxx xxxx	xxxx xx**bb**	**oooo oooo**	Read Device Code **o** at address **b**

Notes: **a** = address high bits
b = address low bits
H = 0 - Low byte, 1 - High Byte
o = data out
i = data in
x = don't care
1 = lock bit 1
2 = lock bit 2

Programming Characteristics

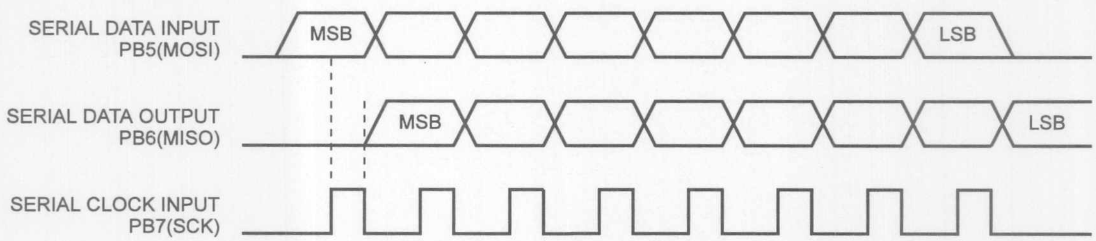


Figure 65. Serial Downloading Waveforms

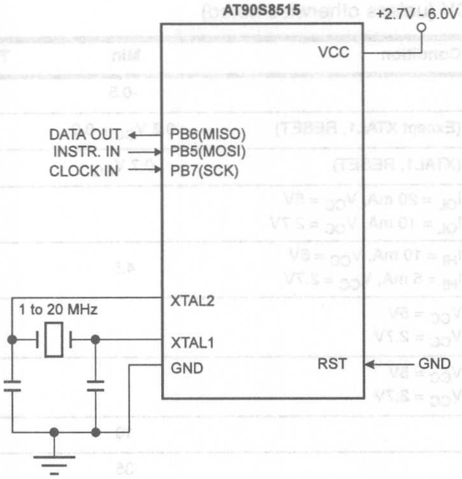


Figure 66. Serial Programming and Verify

When *writing* serial data to the AT90S8515, data is clocked on the *rising* edge of CLK.

When *reading* data from the AT90S8515, data is clocked on the *falling* edge of CLK. See Figure 65 for an explanation.

Absolute Maximum Ratings*

Operating Temperature	-40°C to +105°C
Storage Temperature	-65°C to +150°C
Voltage on any Pin except $\overline{\text{RESET}}$ with respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
I/O Pin Maximum Current	40.0 mA
Maximum Current VCC and GND	140.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

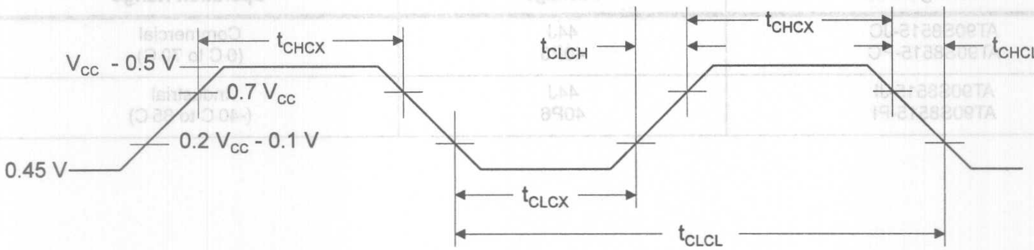
T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage		-0.5		0.2 Vcc - 0.1	V
V_{IH}	Input High Voltage	(Except XTAL1, RESET)	0.2 V_{CC} + 0.9		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RESET)	0.7 V_{CC}		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage (1) (Ports B,C,D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 2.7V			0.5	V
V_{OH}	Output High Voltage (Ports B,C,D)	I_{HI} = 10 mA, V_{CC} = 5V I_{HI} = 5 mA, V_{CC} = 2.7V	4.5			V
I_{OH}	Output Source Current (Ports B,C,D)	V_{CC} = 5V V_{CC} = 2.7V			10 5	mA
I_{OL}	Output Sink Current (Port B,C,D)	V_{CC} = 5V V_{CC} = 2.7V			20 10	mA
RRST	Reset Pulldown Resistor		10		50	kΩ
$R_{I/O}$	I/O Pin Pull-Up Resistor		35		120	kΩ
I_{CC}	Power Supply Current	Active Mode, 3V, 4MHz		3.5		mA
		Idle Mode 3V, 4MHz		1000		µA
I_{CC}	Power Down Mode(2)	WDT enabled, 3V		50		µA
		WDT disabled, 3V		<1		µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V			20	mV
I_{ACLK}	Analog Comparator Input Leakage Current		1	5	10	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750 500		ns

Notes:

1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum total I_{OL} for all output pins: 80 mA
   Port A: 26 mA
   Ports A, B, D: 15 mA
   Maximum total I_{OL} for all output pins: 70 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification.
   Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power Down is 2V.

External Clock Drive Waveforms



External Clock Drive

Symbol	Parameter	V_{CC} = 2.7V to 6.0V		V_{CC} = 4.0V to 6.0V		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency	0	10	0	20	MHz
t_{CLCL}	Clock Period	100		41.7		ns
t_{CHCX}	High Time	40		16.7		ns
t_{CLCX}	Low Time	40		16.7		ns
t_{CLCH}	Rise Time		10		4.15	ns
t_{CHCL}	Fall Time		10		4.15	ns

Ordering Information

Ordering Code	Package	Operation Range
AT90S8515-JC AT90S8515-PC	44J 40P6	Commercial (0°C to 70°C)
AT90S8515-JI AT90S8515-PI	44J 40P6	Industrial (-40°C to 85°C)

Package Type	
44J	44 Lead, Plastic J-Leaded Chip Carrier (PLCC)
40P6	40 Lead, 0.600" Wide, Plastic Dual in Line Package (PDIP)

AT90S8515 Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	5-23
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	5-24
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	5-24
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-	5-29
$3A ($5A)	GIFR	INTF1	INTF0							5-29
$39 ($59)	TIMSK	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-	5-29
$38 ($58)	TIFR	TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	-	5-30
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00	5-31
$34 ($54)	Reserved									
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	5-34
$32 ($52)	TCNT0	Timer/Counter0 (8 Bit)								5-35
$31 ($51)	Reserved									
$30 ($50)	Reserved									
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	5-37
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	5-38
$2D ($4D)	TCNT1H	Timer/Counter1 - Counter Register High Byte								5-39
$2C ($4C)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								5-39
$2B ($4B)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								5-40
$2A ($4A)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								5-40
$29 ($49)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								5-40
$28 ($48)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								5-40
$27 ($47)	Reserved									
$26 ($46)	Reserved									
$25 ($45)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								5-40
$24 ($44)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								5-40
$23 ($43)	Reserved									
$22 ($42)	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	5-43
$20 ($40)	Reserved									
$1F ($3F)	Reserved									
$1E ($3E)	EEARL	EEPROM Address Register								5-44
$1D ($3D)	EEDR	EEPROM Data Register								5-44
$1C ($3C)	EECR	-	-	-	-	-	EEMWE	EEWE	EERE	5-45
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	5-59
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	5-59
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	5-59
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	5-61
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	5-61
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	5-61
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	5-66
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	5-66
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	5-66
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	5-68
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	5-68
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	5-68
$0F ($2F)	SPDR	SPI Data Register								5-50
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-	5-49
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	5-49
$0C ($2C)	UDR	UART I/O Data Register								5-53
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-	5-53
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	5-54
$09 ($29)	UBRR	UART Baud Rate Register								5-56
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	5-57
…	Reserved									
$00 ($20)	Reserved									

AT90S8515 Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	3
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	3
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	3
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1

Instruction Set Nomenclature:

Status Register (SREG):
SREG: Status register
C: Carry flag in status register
Z: Zero flag in status register
N: Negative flag in status register
V: Twos complement overflow indicator
S: N ⊕ V, For signed tests
H: Half Carry flag in the status register
T: Transfer bit used by BLD and BST instructions
I: Global interrupt enable/disable flag

Registers and operands:
Rd: Destination (and source) register in the register file
Rr: Source register in the register file
R: Result after instruction is executed
K: Constant literal or byte data (8 bit)
k: Constant address data for program counter
b: Bit in the register file or I/O register (3 bit)
s: Bit in the status register (3 bit)

X,Y,Z: Indirect address register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
P: I/O port address
q: Displacement for direct addressing (6 bit)

I/O Registers
RAMPX, RAMPY, RAMPZ: Registers concatenated with the X, Y and Z registers enabling indirect addressing of the whole SRAM area on MCUs with more than 64K bytes SRAM.

Stack:
STACK: Stack for return address and pushed registers
SP: Stack Pointer to STACK

Flags:
⇔: Flag affected by instruction
0: Flag cleared by instruction
1: Flag set by instruction
-: Flag not affected by instruction

Conditional Branch Summary

Test	Boolean	Mnemonic	Complementary	Boolean	Mnemonic	Comment
Rd > Rr	Z.(N ⊕ V) = 0	BRLT*	Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Signed
Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Rd < Rr	(N ⊕ V) = 1	BRLT	Signed
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Signed
Rd ≤ Rr	Z+(N ⊕ V) = 1	BRGE*	Rd > Rr	Z.(N ⊕ V) = 0	BRLT*	Signed
Rd < Rr	(N ⊕ V) = 1	BRLT	Rd ≥ Rr	(N ⊕ V) = 0	BRGE	Signed
Rd > Rr	C + Z = 0	BRLO*	Rd ≤ Rr	C + Z = 1	BRSH*	Unsigned
Rd ≥ Rr	C = 0	BRSH/BRCC	Rd < Rr	C = 1	BRLO/BRCS	Unsigned
Rd = Rr	Z = 1	BREQ	Rd ≠ Rr	Z = 0	BRNE	Unsigned
Rd ≤ Rr	C + Z = 1	BRSH*	Rd > Rr	C + Z = 0	BRLO*	Unsigned
Rd < Rr	C = 1	BRLO/BRCS	Rd ≥ Rr	C = 0	BRSH/BRCC	Unsigned
Carry	C = 1	BRCS	No carry	C = 0	BRCC	Simple
Negative	N = 1	BRMI	Positive	N = 0	BRPL	Simple
Overflow	V = 1	BRVS	No overflow	V = 0	BRVC	Simple
Zero	Z = 1	BREQ	Not zero	Z = 0	BRNE	Simple

* Interchange Rd and Rr in the operation before the test. i.e. CP Rd,Rr → CP Rr,Rd

6

0856A

Complete Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rd, K	Add Immediate to Word	Rd+1:Rd ← Rd+1:Rd + K	Z,C,N,V	2
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rd, K	Subtract Immediate from Word	Rd+1:Rd ← Rd+1:Rd - K	Z,C,N,V	2
AND	Rd, Rr	Logical AND	Rd ← Rd . Rr	Z,N,V	1
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd . K	Z,N,V	1
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd . ($FFh - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd . Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,H,	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,H	1
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,H	1

√) Not available in base-line microcontrollers

(continued)

Complete Instruction Set Summary (continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Jump	PC ← k	None	3
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Call Subroutine	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if(I/O(P,b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register Set	If(I/O(P,b)=1) PC← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC+ k + 1	None	1 / 2
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1 / 2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1 / 2

(continued)

6

Complete Instruction Set Summary (continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	3
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Increment	Rd ← (X), X ← X + 1	None	2
LD	Rd, -X	Load Indirect and Pre-Decrement	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, -Y	Load Indirect and Pre-Decrement	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
STS	k, Rr	Store Direct to SRAM	Rd ← (k)	None	3
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr, X ← X + 1	None	2
ST	-X, Rr	Store Indirect and Pre-Decrement	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr, Y ← Y + 1	None	2
ST	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2

(continued)

Complete Instruction Set Summary (continued)

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
BIT AND BIT-TEST INSTRUCTIONS					
LSL	Rd	Logical Shift Left	Rd(n+1)←Rd(n),Rd(0)←0,C←Rd(7	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n)←Rd(n+1),Rd(7)←0,C←Rd(0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)←Rd(n),C←Rd(7	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)←Rd(n+1),C←Rd(0	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	P, b	Set Bit in I/O Register	I/O(P, b) ← 1	None	2
CBI	P, b	Clear Bit in I/O Register	I/O(P, b) ← 0	None	2
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Two's Complement Overflow	V ← 1	V	1
CLV		Clear Two's Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR)	None	1

6

ADC - Add with Carry

Description:

Adds two registers and the contents of the C flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd + Rr + C

	Syntax:	Operands:	Program Counter:
(i)	ADC Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0001	11rd	dddd	rrrr

Status Register (SREG) Boolean Formulae:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 + \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{Rd7} \bullet \overline{Rr7} \bullet \overline{Rr7} \bullet \overline{R7} \bullet \overline{R7} \bullet \overline{Rd7}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
                    ; Add R1:R0 to R3:R2
add    r2,r0        ; Add low byte
adc    r3,r1        ; Add with carry high byte
```

Words: 1 (2 bytes)
Cycles: 1

ADD - Add without Carry

Description:

Adds two registers without the C flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd + Rr$

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADD Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 + \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
add   r1,r2     ; Add r2 to r1 (r1=r1+r2)
add   r28,r28   ; Add r28 to itself (r28=r28+r28)
```

Words: 1 (2 bytes)
Cycles: 1

ADIW - Add Immediate to Word

Description:

Adds an immediate value (0-63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the pointer registers.

Operation:

(i) Rdh:Rdl ← Rdh:Rdl + K

	Syntax:	Operands:	Program Counter:
(i)	ADIW Rdl,K	dl ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16 bit Opcode:

1001	0110	KKdd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: Rdh7 R15
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $0000; cleared otherwise.

C: $\overline{R15} \bullet Rdh7$
Set if there was carry from the MSB of the result; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
adiw  r24,1       ; Add 1 to r25:r24
adiw  r30,63      ; Add 63 to the Z pointer(r31:r30)
```

Words: 1 (2 bytes)
Cycles: 2

AND - Logical AND

Description:

Performs the logical AND between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \bullet Rr$

	Syntax:	Operands:	Program Counter:
(i)	AND Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

0010	00rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
and  r2,r3      ; Bitwise and r2 and r3, result in r2
ldi  r16,1      ; Set bitmask 0000 0001 in r16
and  r2,r16     ; Isolate bit 0 in r2
```

Words: 1 (2 bytes)
Cycles: 1

6

ANDI - Logical AND with Immediate

Description:

Performs the logical AND between the contents of register Rd and a constant and places the result in the destination register Rd.

	Operation:		
(i)	Rd ← Rd • K		
	Syntax:	**Operands:**	**Program Counter:**
(i)	ANDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16 bit Opcode:

0111	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: N ⊕ V, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
andi  r17,$0F     ; Clear upper nibble of r17
andi  r18,$10     ; Isolate bit 4 in r18
andi  r19,$AA     ; Clear odd bits of r19
```

Words: 1 (2 bytes)
Cycles: 1

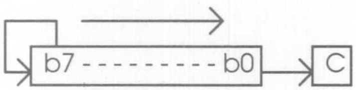
ASR - Arithmetic Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C flag of the SREG. This operation effectively divides a twos complement value by two without changing its sign. The carry flag can be used to round the result.

Operation:

(i)



	Syntax:	**Operands:**	**Program Counter:**
(i)	ASR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	010d	dddd	0101

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); Cleared otherwise (for values of N and C after the shift).

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: Rd0
Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
ldi   r16,$10      ; Load decimal 16 into r16
asr   r16          ; r16=r16 / 2
ldi   r17,$FC      ; Load -4 in r17
asr   r17          ; r17=r17/2
```

Words: 1 (2 bytes)
Cycles: 1

BCLR - Bit Clear in SREG

Description:

Clears a single flag in SREG.

	Operation:
(i)	SREG(s) ← 0

	Syntax:	**Operands:**	**Program Counter:**
(i)	BCLR s	0 ≤ s ≤ 7	PC ← PC + 1

16 bit Opcode:

1001	0100	1sss	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

I: 0 if s = 7; Unchanged otherwise.

T: 0 if s = 6; Unchanged otherwise.

H: 0 if s = 5; Unchanged otherwise.

S: 0 if s = 4; Unchanged otherwise.

V: 0 if s = 3; Unchanged otherwise.

N: 0 if s = 2; Unchanged otherwise.

Z: 0 if s = 1; Unchanged otherwise.

C: 0 if s = 0; Unchanged otherwise.

Example:

```
bclr   0        ; Clear carry flag
bclr   7        ; Disable interrupts
```

Words: 1 (2 bytes)
Cycles: 1

BLD - Bit Load from the T Flag in SREG to a Bit in Register.

Description:

Copies the T flag in the SREG (status register) to bit b in register Rd.

Operation:

(i) Rd(b) ← T

	Syntax:	Operands:	Program Counter:
(i)	BLD Rd,b	$0 \le d \le 31$, $0 \le b \le 7$	PC ← PC + 1

16 bit Opcode:

1111	100d	dddd	0bbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
                ; Copy bit
bst   r1,2      ; Store bit 2 of r1 in T flag
bld   r0,4      ; Load T flag into bit 4 of r0
```

Words: 1 (2 bytes)
Cycles: 1

BRBC - Branch if Bit in SREG is Cleared

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRBC s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	ksss

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
cpi   r20,5      ; Compare r20 to the value 5
brbc  1,noteq    ; Branch if zero flag cleared
...
noteq:nop        ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRBS - Branch if Bit in SREG is Set

Description:

Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form.

Operation:

(i) If SREG(s) = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRBS s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	ksss

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        bst   r0,3      ; Load T bit with bit 3 of r0
        brbs  6,bitset  ; Branch T bit was set
        ...
bitset: nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRCC - Branch if Carry Cleared

Description:

Conditional relative branch. Tests the Carry flag (C) and branches relatively to PC if C is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If C = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRCC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
          addr22,r23    ; Add r23 to r22
          brccnocarry   ; Branch if carry cleared
          ...
nocarry:  nop           ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRCS - Branch if Carry Set

Description:

Conditional relative branch. Tests the Carry flag (C) and branches relatively to PC if C is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If C = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRCS k	$-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        cpi   r26,$56   ; Compare r26 with $56
        brcs  carry     ; Branch if carry set
        ...
carry:  nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BREQ - Branch if Equal

Description:

Conditional relative branch. Tests the Zero flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k).

Operation:

(i) If Rd = Rr (Z = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BREQ k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k001

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        cpr1,r0      ; Compare registers r1 and r0
        breqequal    ; Branch if registers equal
        ...
equal:  nop          ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRGE - Branch if Greater or Equal (Signed)

Description:

Conditional relative branch. Tests the Signed flag (S) and branches relatively to PC if S is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was greater than or equal to the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 4,k).

Operation:

(i) If Rd ≥ Rr ($N \oplus V = 0$) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRGE k	$-64 \leq k \leq +63$	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k100

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
          cpr11,r12    ; Compare registers r11 and r12
          brgegreateq  ; Branch if r11 >= r12 (signed)
          ...
greateq:  nop          ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRHC - Branch if Half Carry Flag is Cleared

Description:

Conditional relative branch. Tests the Half Carry flag (H) and branches relatively to PC if H is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 5,k).

Operation:

(i) If H = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRHC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k101

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        brhc hclear    ; Branch if half carry flag cleared
        ...
hclear: nop            ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRHS - Branch if Half Carry Flag is Set

Description:
Conditional relative branch. Tests the Half Carry flag (H) and branches relatively to PC if H is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 5,k).

Operation:

(i) If H = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRHS k	$-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k101

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        brhshset    ; Branch if half carry flag set
        ...
hset:   nop         ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true

6

BRID - Branch if Global Interrupt is Disabled

Description:

Conditional relative branch. Tests the Global Interrupt flag (I) and branches relatively to PC if I is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 7,k).

Operation:

(i) If I = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRID k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
          brid intdis    ; Branch if interrupt disabled
          ...
intdis:   nop            ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRIE - Branch if Global Interrupt is Enabled

Description:

Conditional relative branch. Tests the Global Interrupt flag (I) and branches relatively to PC if I is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 7,k).

Operation:

(i) If I = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRIE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
            brieinten     ; Branch if interrupt enabled
            ...
    inten:  nop           ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRLO - Branch if Lower (Unsigned)

Description:

Conditional relative branch. Tests the Carry flag (C) and branches relatively to PC if C is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned binary number represented in Rd was smaller than the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 0,k).

Operation:

(i) If Rd < Rr (C = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRLO k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        eor   r19,r19    ; Clear r19
loop:   inc   r19        ; Increase r19
        ...
        cpi   r19,$10    ; Compare r19 with $10
        brlo  loop       ; Branch if r19 < $10 (unsigned)
        nop              ; Exit from loop (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1 if condition is false
2 if condition is true

BRLT - Branch if Less Than (Signed)

Description:

Conditional relative branch. Tests the Signed flag (S) and branches relatively to PC if S is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the signed binary number represented in Rd was less than the signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 4,k).

Operation:

(i) If Rd < Rr ($N \oplus V = 1$) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRLT k	$-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k100

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        cp    r16,r1      ; Compare r16 to r1
        brlt  less        ; Branch if r16 < r1 (signed)
        ...
less:   nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

6

BRMI - Branch if Minus

Description:

Conditional relative branch. Tests the Negative flag (N) and branches relatively to PC if N is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 2,k).

Operation:

(i) If N = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRMI k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k010

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
            subi   r18,4        ; Subtract 4 from r18
            brmi   negative     ; Branch if result negative
            ...
negative:   nop                 ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRNE - Branch if Not Equal

Description:

Conditional relative branch. Tests the Zero flag (Z) and branches relatively to PC if Z is cleared. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was not equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 1,k).

Operation:

(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRNE k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k001

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        eor    r27,r27    ; Clear r27
loop:   inc    r27        ; Increase r27
        ...
        cpi    r27,5      ; Compare r27 to 5
        brne   loop       ; Branch if r27<>5
        nop               ; Loop exit (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRPL - Branch if Plus

Description:

Conditional relative branch. Tests the Negative flag (N) and branches relatively to PC if N is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 2,k).

Operation:

(i) If N = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRPL k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k010

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
            subi  r26,$50     ; Subtract $50 from r26
            brpl  positive    ; Branch if r26 positive
            ...
positive:   nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRSH - Branch if Same or Higher (Unsigned)

Description:

Conditional relative branch. Tests the Carry flag (C) and branches relatively to PC if C is cleared. If the instruction is executed immediately after execution of any of the instructions CP, CPI, SUB or SUBI the branch will occur if and only if the unsigned binary number represented in Rd was greater than or equal to the unsigned binary number represented in Rr. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 0,k).

Operation:

(i) If Rd ≥Rr (C = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRSH k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        subi  r19,4        ; Subtract 4 from r19
        brsh  highsm       ; Branch if r19 >= 4 (unsigned)
        ...
highsm: nop                ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRTC - Branch if the T Flag is Cleared

Description:

Conditional relative branch. Tests the T flag and branches relatively to PC if T is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 6,k).

Operation:

(i) If T = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRTC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k110

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        bst     r3,5      ; Store bit 5 of r3 in T flag
        brtc    tclear    ; Branch if this bit was cleared
        ...
tclear: nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRTS - Branch if the T Flag is Set

Description:

Conditional relative branch. Tests the T flag and branches relatively to PC if T is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 6,k).

Operation:

(i) If T = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRTS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k110

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        bst   r3,5      ; Store bit 5 of r3 in T flag
        brts  tset      ; Branch if this bit was set
        ...
tset:   nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRVC - Branch if Overflow Cleared

Description:

Conditional relative branch. Tests the Overflow flag (V) and branches relatively to PC if V is cleared. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBC 3,k).

Operation:

(i) If V = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRVC k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	01kk	kkkk	k011

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
         add  r3,r4      ; Add r4 to r3
         brvc noover     ; Branch if no overflow
         ...
noover:  nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BRVS - Branch if Overflow Set

Description:

Conditional relative branch. Tests the Overflow flag (V) and branches relatively to PC if V is set. This instruction branches relatively to PC in either direction (PC-64≤destination≤PC+63). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 3,k).

Operation:

(i) If V = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRVS k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k011

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        add   r3,r4    ; Add r4 to r3
        brvs  overfl   ; Branch if overflow
        ...
overfl: nop            ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true

BSET - Bit Set in SREG

Description:

Sets a single flag or bit in SREG.

	Operation:
(i)	SREG(s) ← 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	BSET s	$0 \le s \le 7$	PC ← PC + 1

16 bit Opcode:

1001	0100	0sss	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

I: 1 if s = 7; Unchanged otherwise.

T: 1 if s = 6; Unchanged otherwise.

H: 1 if s = 5; Unchanged otherwise.

S: 1 if s = 4; Unchanged otherwise.

V: 1 if s = 3; Unchanged otherwise.

N: 1 if s = 2; Unchanged otherwise.

Z: 1 if s = 1; Unchanged otherwise.

C: 1 if s = 0; Unchanged otherwise.

Example:

```
bset    6        ; Set T flag
bset    7        ; Enable interrupt
```

Words: 1 (2 bytes)
Cycles: 1

BST - Bit Store from Bit in Register to T Flag in SREG

Description:

Stores bit b from Rd to the T flag in SREG (status register).

Operation:

(i) T ← Rd(b)

	Syntax:	Operands:	Program Counter:
(i)	BST Rd,b	$0 \le d \le 31, 0 \le b \le 7$	PC ← PC + 1

16 bit Opcode:

1111	101d	dddd	Xbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	⇔	-	-	-	-	-	-

T: 0 if bit b in Rd is cleared. Set to 1 otherwise.

Example:

```
                         ; Copy bit
    bst     r1,2         ; Store bit 2 of r1 in T flag
    bld     r0,4         ; Load T into bit 4 of r0
```

Words: 1 (2 bytes)
Cycles: 1

6

CALL - Long Call to a Subroutine

Description:

Calls to a subroutine within the entire program memory. The return address (to the instruction after the CALL) will be stored onto the stack. (See also RCALL).

	Operation:	
(i)	PC ← k	Devices with 16 bits PC, 128K bytes program memory maximum.
(ii)	PC ← k	Devices with 22 bits PC, 8M bytes program memory maximum.

	Syntax:	**Operands:**	**Program Counter:Stack**
(i)	CALL k	$0 \leq k \leq 64K$	PC ← kSTACK ← PC+2 SP ← SP-2, (2 bytes, 16 bits)
(ii)	CALL k	$0 \leq k \leq 4M$	PC ← kSTACK ← PC+2 SP ← SP-3 (3 bytes, 22 bits)

32 bit Opcode:

1001	010k	kkkk	111k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        mov     r16,r0          ; Copy r0 to r16
        call    check           ; Call subroutine
        nop                     ; Continue (do nothing)
        ...
check:  cpi     r16,$42         ; Check if r16 has a special value
        breq    error           ; Branch if equal
        ret                     ; Return from subroutine
        ...
error:  rjmp    error           ; Infinite loop
```

Words: 2 (4 bytes)
Cycles: 4

CBI - Clear Bit in I/O Register

Description:

Clears a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers - addresses 0-31.

Operation:

(i) I/O(P,b) ← 0

	Syntax:	Operands:	Program Counter:
(i)	CBI P,b	$0 \le P \le 31$, $0 \le b \le 7$	PC ← PC + 1

16 bit Opcode:

1001	1000	pppp	pbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
cbi     $12,7          ; Clear bit 7 in Port D
```

Words: 1 (2 bytes)

Cycles: 2

CBR - Clear Bits in Register

Description:

Clears the specified bits in register Rd. Performs the logical AND between the contents of register Rd and the complement of the constant mask K. The result will be placed in register Rd.

	Operation:		
(i)	Rd ← Rd • ($FF - K)		
	Syntax:	**Operands:**	**Program Counter:**
(i)	CBR Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	PC ← PC + 1

16 bit Opcode: See ANDI with K complemented.

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
cbr    r16,$F0    ; Clear upper nibble of r16
cbr    r18,1      ; Clear bit 0 in r18
```

Words: 1 (2 bytes)
Cycles: 1

CLC - Clear Carry Flag

Description:

Clears the Carry flag (C) in SREG (status register).

Operation:

(i) C ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLC	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1000	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	0

C: 0

Carry flag cleared

Example:

```
add   r0,r0    ; Add r0 to itself
clc            ; Clear carry flag
```

Words: 1 (2 bytes)

Cycles: 1

CLH - Clear Half Carry Flag

Description:

Clears the Half Carry flag (H) in SREG (status register).

	Operation:		
(i)	$H \leftarrow 0$		
	Syntax:	**Operands:**	**Program Counter:**
(i)	CLH	None	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	0100	1101	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	0	-	-	-	-	-

H: 0
Half Carry flag cleared

Example:

```
clh          ; Clear the Half Carry flag
```

Words: 1 (2 bytes)
Cycles: 1

CLI - Clear Global Interrupt Flag

Description:

Clears the Global Interrupt flag (I) in SREG (status register).

Operation:

(i) I ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLI	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1111	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
0	-	-	-	-	-	-	-

I: 0
Global Interrupt flag cleared

Example:

```
cli             ; Disable interrupts
in    r11,$16   ; Read port B
sei             ; Enable interrupts
```

Words: 1 (2 bytes)
Cycles: 1

6

CLN - Clear Negative Flag

Description:

Clears the Negative flag (N) in SREG (status register).

	Operation:
(i)	N ← 0

	Syntax:	**Operands:**	**Program Counter:**
(i)	CLN	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1010	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	0	-	-

N: 0
Negative flag cleared

Example:

```
add     r2,r3      ; Add r3 to r2
cln                ; Clear negative flag
```

Words: 1 (2 bytes)
Cycles: 1

CLR - Clear Register

Description:

Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear all bits in the register.

	Operation:
(i)	$Rd \leftarrow Rd \oplus Rd$

	Syntax:	Operands:	Program Counter:
(i)	CLR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	0	0	0	1	-

S: 0
Cleared

V: 0
Cleared

N: 0
Cleared

Z: 1
Set

R (Result) equals Rd after the operation.

Example:

```
        clr   r18         ; clear r18
loop:   inc   r18         ; increase r18
        ...
        cpi   r18,$50     ; Compare r18 to $50
        brne  loop
```

Words: 1 (2 bytes)
Cycles: 1

6

CLS - Clear Signed Flag

Description:

Clears the Signed flag (S) in SREG (status register).

	Operation:		
(i)	S ← 0		
	Syntax:	**Operands:**	**Program Counter:**
(i)	CLS	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1100	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	0	-	-	-	-

S: 0
Signed flag cleared

Example:

```
add   r2,r3      ; Add r3 to r2
cls              ; Clear signed flag
```

Words: 1 (2 bytes)
Cycles: 1

CLT - Clear T Flag

Description:

Clears the T flag in SREG (status register).

Operation:

(i) T ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLT	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1110	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	0	-	-	-	-	-	-

T: 0
T flag cleared

Example:

```
clt                 ; Clear T flag
```

Words: 1 (2 bytes)

Cycles: 1

CLV - Clear Overflow Flag

Description:

Clears the Overflow flag (V) in SREG (status register).

	Operation:		
(i)	$V \leftarrow 0$		
	Syntax:	**Operands:**	**Program Counter:**
(i)	CLV	None	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	0100	1011	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	0	-	-	-

V: 0
Overflow flag cleared

Example:

```
add     r2,r3      ; Add r3 to r2
clv                ; Clear overflow flag
```

Words: 1 (2 bytes)

Cycles: 1

CLZ - Clear Zero Flag

Description:

Clears the Zero flag (Z) in SREG (status register).

Operation:

(i) Z ← 0

	Syntax:	Operands:	Program Counter:
(i)	CLZ	None	PC ← PC + 1

16 bit Opcode:

1001	0100	1001	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	0	-

Z: 0
Zero flag cleared

Example:

```
add    r2,r3     ; Add r3 to r2
clz              ; Clear zero
```

Words: 1 (2 bytes)
Cycles: 1

COM - One's Complement

Description:

This instruction performs a one's complement of register Rd.

Operation:

(i) Rd ← \$FF - Rd

	Syntax:	Operands:	Program Counter:
(i)	COM Rd	$0 \le d \le 31$	PC ← PC + 1

16 bit Opcode:

1001	010d	dddd	0000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	1

S: $N \oplus V$
For signed tests.

V: 0
Cleared.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; Cleared otherwise.

C: 1
Set.

R (Result) equals Rd after the operation.

Example:

```
        com     r4          ; Take one's complement of r4
        breq    zero        ; Branch if zero
        ...
zero:   nop                 ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

CP - Compare

Description:

This instruction performs a compare between two registers Rd and Rr. None of the registers are changed. All conditional branches can be used after this instruction.

	Operation:
(i)	Rd - Rr

	Syntax:	**Operands:**	**Program Counter:**
(i)	CP Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0001	01rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet Rr3 + Rr3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{Rd7} \bullet R7 + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $Rd7 \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet Rd7$
Set if the result is \$00; cleared otherwise.

C: $\overline{Rd7} \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
        cp    r4,r19      ; Compare r4 with r19
        brne  noteq       ; Branch if r4 <> r19
        ...
noteq:  nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

CPC - Compare with Carry

Description:

This instruction performs a compare between two registers Rd and Rr and also takes into account the previous carry. None of the registers are changed. All conditional branches can be used after this instruction.

	Operation:		
(i)	Rd - Rr - C		
	Syntax:	**Operands:**	**Program Counter:**
(i)	CPC Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0000	01rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet Rr3 + Rr3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{Rr7} \bullet \overline{R7} + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0} \bullet Z$
Previous value remains unchanged when the result is zero; cleared otherwise.

C: $\overline{Rd7} \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
                                    ; Compare r3:r2 with r1:r0
        cp      r2,r0               ; Compare low byte
        cpc     r3,r1               ; Compare high byte
        brne    noteq               ; Branch if not equal
        ...
noteq:  nop                         ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

CPI - Compare with Immediate

Description:

This instruction performs a compare between register Rd and a constant. The register is not changed. All conditional branches can be used after this instruction.

Operation:

(i) Rd - K

	Syntax:	Operands:	Program Counter:
(i)	CPI Rd,K	$16 \leq d \leq 31$, $0 \leq K \leq 255$	PC ← PC + 1

16 bit Opcode:

0011	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet K3 + K3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{K7} \bullet \overline{R7} + \overline{Rd7} \bullet K7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $\overline{Rd7} \bullet K7 + K7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) after the operation.

Example:

```
        cpi     r19,3      ; Compare r19 with 3
        brne    error      ; Branch if r19<>3
        ...
error:  nop                ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

CPSE - Compare Skip if Equal

Description:
This instruction performs a compare between two registers Rd and Rr, and skips the next instruction if Rd = Rr.

Operation:

(i) If Rd = Rr then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	CPSE Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a two word instruction

16 bit Opcode:

0001	00rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        inc     r4          ; Increase r4
        cpse    r4,r0       ; Compare r4 to r0
        neg     r4          ; Only executed if r4<>r0
        nop                 ; Continue (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

DEC - Decrement

Description:

Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd.

The C flag in SREG is not affected by the operation, thus allowing the DEC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned values, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) $Rd \leftarrow Rd - 1$

	Syntax:	Operands:	Program Counter:
(i)	DEC Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	010d	dddd	1010

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	-

S: $N \oplus V$
For signed tests.

V: $\overline{R7} \bullet R6 \bullet R5 \bullet R4 \bullet R3 \bullet R2 \bullet R1 \bullet R0$
Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was \$80 before the operation.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        ldi   r17,$10      ; Load constant in r17
loop:   add   r1,r2        ; Add r2 to r1
        dec   r17          ; Decrement r17
        brne  loop         ; Branch if r17<>0
        nop                ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

EOR - Exclusive OR

Description:

Performs the logical EOR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \oplus Rr$

	Syntax:	Operands:	Program Counter:
(i)	EOR Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

0010	01rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
eor     r4,r4       ; Clear r4
eor     r0,r22      ; Bitwise exclusive or between r0 and r22
```

Words: 1 (2 bytes)

Cycles: 1

ICALL - Indirect Call to Subroutine

Description:

Indirect call of a subroutine pointed to by the Z (16 bits) pointer register in the register file. The Z pointer register is 16 bits wide and allows call to a subroutine within the current 64K words (128K bytes) section in the program memory space.

Operation:

(i) PC(15-0) ← Z(15 - 0)Devices with 16 bits PC, 128K bytes program memory maximum.

(ii) PC(15-0) ← Z(15 - 0)Devices with 22 bits PC, 8M bytes program memory maximum.
PC(21-16) is unchanged

	Syntax:	Operands:	Program Counter:	Stack
(i)	ICALL	None	See Operation	STACK ← PC+1 SP ← SP-2 (2 bytes, 16 bits)
(ii)	ICALL	None	See Operation	STACK ← PC+1 SP ← SP-3 (3 bytes, 22 bits)

16 bit Opcode:

1001	0101	XXXX	1001

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
mov     r30,r0      ; Set offset to call table
icall               ; Call routine pointed to by r31:r30
```

Words: 1 (2 bytes)

Cycles: 3

IJMP - Indirect Jump

Description:

Indirect jump to the address pointed to by the Z (16 bits) pointer register in the register file. The Z pointer register is 16 bits wide and allows jump within the current 64K words (128K bytes) section of program memory.

	Operation:	
(i)	PC ← Z(15 - 0)	Devices with 16 bits PC, 128K bytes program memory maximum.
(ii)	PC(15-0) ← Z(15-0) PC(21-16) is unchanged	Devices with 22 bits PC, 8M bytes program memory maximum.

	Syntax:	**Operands:**	**Program Counter:**	**Stack**
(ii)	IJMP	None	See Operation	Not Affected
(iii)	IJMP	None	See Operation	Not Affected

16 bit Opcode:

1001	0100	XXXX	1001

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
mov     r30,r0      ; Set offset to jump table
ijmp                ; Jump to routine pointed to by r31:r30
```

Words: 1 (2 bytes)
Cycles: 2

IN - Load an I/O Port to Register

Description:

Loads data from the I/O Space (Ports, Timers, Configuration registers etc.) into register Rd in the register file.

Operation:

(i) Rd ← P

	Syntax:	Operands:	Program Counter:
(i)	IN Rd,P	$0 \le d \le 31, 0 \le P \le 63$	PC ← PC + 1

16 bit Opcode:

1011	0PPd	dddd	PPPP

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        in      r25,$16     ; Read Port B
        cpi     r25,4       ; Compare read value to constant
        breq    exit        ; Branch if r25=4
        ...
exit:   nop                 ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

INC - Increment

Description:

Adds one -1- to the contents of register Rd and places the result in the destination register Rd.

The C flag in SREG is not affected by the operation, thus allowing the INC instruction to be used on a loop counter in multiple-precision computations.

When operating on unsigned numbers, only BREQ and BRNE branches can be expected to perform consistently. When operating on two's complement values, all signed branches are available.

Operation:

(i) $Rd \leftarrow Rd + 1$

	Syntax:	**Operands:**	**Program Counter:**
(i)	INC Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	010d	dddd	0011

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	-

S: $N \oplus V$

For signed tests.

V: $R7 \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was $7F before the operation.

N: R7

Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$

Set if the result is $00; Cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        clr     r22         ; clear r22
loop:   inc     r22         ; increment r22
        ...
        cpi     r22,$4F     ; Compare r22 to $4f
        brne    loop        ; Branch if not equal
        nop                 ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

JMP - Jump

Description:

Jump to an address within the entire 4M (words) program memory. See also RJMP.

Operation:

(i) PC ← k

	Syntax:	Operands:	Program Counter:	Stack
(i)	JMP k	$0 \le k \le 4M$	PC ← k	Unchanged

32 bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        mov     r1,r0       ; Copy r0 to r1
        jmp     farplc      ; Unconditional jump
        ...
farplc: nop                 ; Jump destination (do nothing)
```

Words: 2 (4 bytes)
Cycles: 3

6

LD - Load Indirect from SRAM to Register using Index X

Description:

Loads one byte indirect from SRAM to register. The SRAM location is pointed to by the X (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64K bytes. To access another SRAM page the RAMPX in register in the I/O area has to be changed.

The X pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for accessing arrays, tables, and stack pointer usage of the X pointer register.

Using the X pointer:

	Operation:	Comment:	
(i)	Rd ← (X)		X: Unchanged
(ii)	Rd ← (X)	X ← X + 1	X: Post incremented
(iii)	X ← X - 1	Rd ← (X)	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, X	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, X+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd,-X	$0 \le d \le 31$	PC ← PC + 1

16 bit Opcode :

(i)	1001	000d	dddd	1100
(ii)	1001	000d	dddd	1101
(iii)	1001	000d	dddd	1110

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr  r27      ; Clear X high byte
ldi  r26,$20  ; Set X low byte to $20
ld   r0,X+    ; Load r0 with SRAM loc. $20(X post inc)
ld   r1,X     ; Load r1 with SRAM loc. $21
ldi  r26,$23  ; Set X low byte to $23
ld   r2,X     ; Load r2 with SRAM loc. $23
ld   r3,-X    ; Load r3 with SRAM loc. $22(X pre dec)
```

Words: 1 (2 bytes)
Cycles: 2

LD (LDD) - Load Indirect from SRAM to Register using Index Y

Description:

Loads one byte indirect with or without displacement from SRAM to register. The SRAM location is pointed to by the Y (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64K bytes. To access another SRAM page the RAMPY register in the I/O area has to be changed.

The Y pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for accessing arrays, tables, and stack pointer usage of the Y pointer register.

Using the Y pointer:

	Operation:		Comment:
(i)	$Rd \leftarrow (Y)$		Y: Unchanged
(ii)	$Rd \leftarrow (Y)$	$Y \leftarrow Y + 1$	Y: Post incremented
(iii)	$Y \leftarrow Y - 1$	$Rd \leftarrow (Y)$	Y: Pre decremented
(iiii)	$Rd \leftarrow (Y+q)$		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Y	$0 \le d \le 31$	$PC \leftarrow PC + 1$
(ii)	LD Rd, Y+	$0 \le d \le 31$	$PC \leftarrow PC + 1$
(iii)	LD Rd,-Y	$0 \le d \le 31$	$PC \leftarrow PC + 1$
(iiii)	LDD Rd, Y+q	$0 \le d \le 31$, $0 \le q \le 63$	$PC \leftarrow PC + 1$

16 bit Opcode :

(i)	1000	000d	dddd	1000
(ii)	1001	000d	dddd	1001
(iii)	1001	000d	dddd	1010
(iiii)	10q0	qq0d	dddd	1qqq

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr  r29        ; Clear Y high byte
ldi  r28,$20    ; Set Y low byte to $20
ld   r0,Y+      ; Load r0 with SRAM loc. $20(Y post inc)
ld   r1,Y       ; Load r1 with SRAM loc. $21
ldi  r28,$23    ; Set Y low byte to $23
ld   r2,Y       ; Load r2 with SRAM loc. $23
ld   r3,-Y      ; Load r3 with SRAM loc. $22(Y pre dec)
ldd  r4,Y+2     ; Load r4 with SRAM loc. $24
```

Words: 1 (2 bytes)
Cycles: 2

LD (LDD) - Load Indirect From SRAM to Register using Index Z

Description:

Loads one byte indirectly with or without displacement from SRAM to register. The SRAM location is pointed to by the Z (16 bits) pointer register in the register file. Memory access is limited to the current SRAM page of 64K bytes. To access another SRAM page the RAMPZ register in the I/O area has to be changed.

The Z pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the Z pointer register, however because the Z pointer register can be used for indirect subroutine calls, indirect jumps and table lookup, it is often more convenient to use the X or Y pointer as a dedicated stack pointer.

For using the Z pointer for table lookup in program memory see the LPM instruction.

Using the Z pointer:

	Operation:	Comment:	
(i)	Rd ← (Z)		Z: Unchanged
(ii)	Rd ← (Z)	Z ← Z + 1	Z: Post increment
(iii)	Z ← Z -1	Rd ← (Z)	Z: Pre decrement
(iiii)	Rd ← (Z+q)		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	LD Rd, Z	$0 \le d \le 31$	PC ← PC + 1
(ii)	LD Rd, Z+	$0 \le d \le 31$	PC ← PC + 1
(iii)	LD Rd,-Z	$0 \le d \le 31$	PC ← PC + 1
(iiii)	LDD Rd, Z+q	$0 \le d \le 31$, $0 \le q \le 63$	PC ← PC + 1

16 bit Opcode :

(i)	1000	000d	dddd	0000
(ii)	1001	000d	dddd	0001
(iii)	1001	000d	dddd	0010
(iiii)	10q0	qq0d	dddd	0qqq

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr   r31        ; Clear Z high byte
ldi   r30,$20    ; Set Z low byte to $20
ld    r0,Z+      ; Load r0 with SRAM loc. $20(Z post inc)
ld    r1,Z       ; Load r1 with SRAM loc. $21
ldi   r30,$23    ; Set Z low byte to $23
ld    r2,Z       ; Load r2 with SRAM loc. $23
ld    r3,-Z      ; Load r3 with SRAM loc. $22(Z pre dec)
ldd   r4,Z+2     ; Load r4 with SRAM loc. $24
```

Words: 1 (2 bytes)

Cycles: 2

LDI - Load Immediate

Description:

Loads an 8 bit constant directly to register 16 to 31.

Operation:

(i) Rd ← K

	Syntax:	Operands:	Program Counter:
(i)	LDI Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	PC ← PC + 1

16 bit Opcode:

1110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr   r31         ; Clear Z high byte
ldi   r30,$F0     ; Set Z low byte to $F0
lpm               ; Load constant from program
                  ; memory pointed to by Z
```

Words: 1 (2 bytes)

Cycles: 1

6

LDS - Load Direct from SRAM

Description:

Loads one byte from the SRAM to a Register. A 16-bit address must be supplied. Memory access is limited to the current SRAM Page of 64K bytes. The LDS instruction uses the RAMPZ register to access memory above 64K bytes.

Operation:

(i) Rd ← (k)

	Syntax:	Operands:	Program Counter:
(i)	LDS Rd,k	$0 \le d \le 31$, $0 \le k \le 65535$	PC ← PC + 2

32 bit Opcode:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
lds   r2,$FF00     ; Load r2 with the contents of SRAM location $FF00
add   r2,r1        ; add r1 to r2
sts   $FF00,r2     ; Write back
```

Words: 2 (4 bytes)
Cycles: 3

LPM - Load Program Memory

Description:

Loads one byte pointed to by the Z register into register 0 (R0). This instruction features a 100% space effective constant initialization or constant data fetch. The program memory is organized in 16 bits words and the LSB of the Z (16 bits) pointer selects either low byte (0) or high byte (1). This instruction can address the first 64K bytes (32K words) of program memory.

	Operation:		**Comment:**
(i)	R0 ← (Z)		Z points to program memory

	Syntax:	**Operands:**	**Program Counter:**
(i)	LPM	None	PC ← PC + 1

16 bit Opcode:

1001	0101	110X	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr  r31          ; Clear Z high byte
ldi  r30,$F0      ; Set Z low byte
lpm               ; Load constant from program
                  ; memory pointed to by Z (r31:r30)
```

Words: 1 (2 bytes)

Cycles: 3

LSL - Logical Shift Left

Description:

Shifts all bits in Rd one place to the left. Bit 0 is cleared. Bit 7 is loaded into the C flag of the SREG. This operation effectively multiplies an unsigned value by two.

Operation:

(i)

$$\leftarrow$$

$$\boxed{C} \leftarrow \boxed{b7 \text{-------------------} b0} \leftarrow 0$$

	Syntax:	Operands:	Program Counter:
(i)	LSL Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode: (see ADD Rd,Rd)

0000	11dd	dddd	dddd

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: Rd3

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); Cleared otherwise (for values of N and C after the shift).

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd7
Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
add   r0,r4    ; Add r4 to r0
lsl   r0       ; Multiply r0 by 2
```

Words: 1 (2 bytes)
Cycles: 1

LSR - Logical Shift Right

Description:

Shifts all bits in Rd one place to the right. Bit 7 is cleared. Bit 0 is loaded into the C flag of the SREG. This operation effectively divides an unsigned value by two. The C flag can be used to round the result.

Operation:

$$0 \rightarrow \boxed{b7 \text{ - - - - - - - - - - - - - - - - - - } b0} \rightarrow \boxed{C}$$

	Syntax:	Operands:	Program Counter:
(i)	LSR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	010d	dddd	0110

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	0	⇔	⇔

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); Cleared otherwise (for values of N and C after the shift).

N: 0

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd0
Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
add     r0,r4        ; Add r4 to r0
lsr     r0           ; Divide r0 by 2
```

Words: 1 (2 bytes)
Cycles: 1

MOV - Copy Register

Description:

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

	Operation:
(i)	Rd ← Rr

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0010	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        mov     r16,r0      ; Copy r0 to r16
        call    check       ; Call subroutine
        ...
check:  cpi     r16,$11     ; Compare r16 to $11
        ...
        ret                 ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 1

MUL - Multiply

Description:

This instruction performs 8-bit × 8-bit → 16-bit unsigned multiplication.

Rr		Rd		R1	R0
Multiplicand	×	Multiplier	→	Product High	Product Low
8		8		16	

The multiplicand Rr and the multiplier Rd are two registers. The 16-bit product is placed in R1 (high byte) and R0 (low byte). Note that if the multiplicand and the multiplier is selected from R0 or R1 the result will overwrite those after multiplication.

Operation:

(i) R1,R0 ← Rr × Rd

	Syntax:	Operands:	Program Counter:
(i)	MUL Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

1001	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	⇔

C: R15
Set if bit 15 of the result is set; cleared otherwise.

R (Result) equals R1,R0 after the operation.

Example:

```
mulr6,r5; Multiply r6 and r5
movr6,r1; Copy result back in r6:r5
movr5,r0; Copy result back in r6:r5
```

Words: 1 (2 bytes)
Cycles: 2

Not available in base-line microcontrollers.

NEG - Two's Complement

Description:

Replaces the contents of register Rd with its two's complement; the value $80 is left unchanged.

Operation:

(i) Rd ← $00 - Rd

	Syntax:	Operands:	Program Counter:
(i)	NEG Rd	$0 \le d \le 31$	PC ← PC + 1

16 bit Opcode:

1001	010d	dddd	0001

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$
For signed tests.

V: $R7 \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if there is a two's complement overflow from the implied subtraction from zero; cleared otherwise. A two's complement overflow will occur if and only if the contents of the Register after operation (Result) is $80.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; Cleared otherwise.

C: R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0
Set if there is a borrow in the implied subtraction from zero; cleared otherwise. The C flag will be set in all cases except when the contents of Register after operation is $00.

R (Result) equals Rd after the operation.

Example:

```
          sub   r11,r0      ; Subtract r0 from r11
          brpl  positive    ; Branch if result positive
          neg   r11         ; Take two's complement of r11
positive: nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

NOP - No Operation

Description:

This instruction performs a single cycle No Operation.

Operation:

(i) No

	Syntax:	Operands:	Program Counter:
(i)	NOP	None	PC ← PC + 1

16 bit Opcode:

0000	0000	0000	0000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr   r16      ; Clear r16
ser   r17      ; Set r17
out   $18,r16  ; Write zeros to Port B
nop            ; Wait (do nothing)
out   $18,r17  ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

6

OR - Logical OR

Description:

Performs the logical OR between the contents of register Rd and register Rr and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd v Rr

	Syntax:	Operands:	Program Counter:
(i)	OR Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0010	10rd	dddd	rrrr

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        or      r15,r16     ; Do bitwise or between registers
        bst     r15,6       ; Store bit 6 of r15 in T flag
        brts    ok          ; Branch if T flag set
        ...
ok:     nop                 ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

ORI - Logical OR with Immediate

Description:

Performs the logical OR between the contents of register Rd and a constant and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \lor K$

	Syntax:	Operands:	Program Counter:
(i)	ORI Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	$PC \leftarrow PC + 1$

16 bit Opcode:

0110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
ori   r16,$F0      ; Set high nibble of r16
ori   r17,1        ; Set bit 0 of r17
```

Words: 1 (2 bytes)
Cycles: 1

6

OUT - Store Register to I/O port

Description:

Stores data from register Rr in the register file to I/O space (Ports, Timers, Configuration registers etc.).

Operation:

(i) $P \leftarrow Rr$

	Syntax:	Operands:	Program Counter:
(i)	OUT P,Rr	$0 \le r \le 31$, $0 \le P \le 63$	$PC \leftarrow PC + 1$

16 bit Opcode:

1011	1PPr	rrrr	PPPP

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr  r16          ; Clear r16
ser  r17          ; Set r17
out  $18,r16      ; Write zeros to Port B
nop               ; Wait (do nothing)
out  $18,r17      ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

POP - Pop Register from Stack

Description:

This instruction loads register Rd with a byte from the STACK.

Operation:

(i) Rd ← STACK

	Syntax:	Operands:	Program Counter:Stack
(i)	POP Rd	$0 \le d \le 31$	PC ← PC + 1SP ← SP + 1

16 bit Opcode:

1001	000d	dddd	1111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
          call    routine    ; Call subroutine
          ...
routine:  push    r14        ; Save r14 on the stack
          push    r13        ; Save r13 on the stack
          ...
          pop     r13        ; Restore r13
          pop     r14        ; Restore r14
          ret                ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 2

PUSH - Push Register on Stack

Description:

This instruction stores the contents of register Rr on the STACK.

Operation:

(i) STACK ← Rr

	Syntax:	Operands:	Program Counter:Stack:
(i)	PUSH Rr	$0 \le r \le 31$	PC ← PC + 1SP ← SP - 1

16 bit Opcode:

1001	001d	dddd	1111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
          call    routine   ; Call subroutine
          ...
routine:  push    r14       ; Save r14 on the stack
          push    r13       ; Save r13 on the stack
          ...
          pop     r13       ; Restore r13
          pop     r14       ; Restore r14
          ret               ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 2

RCALL - Relative Call to Subroutine

Description:

Calls a subroutine within ± 2K words (4K bytes). The return address (the instruction after the RCALL) is stored onto the stack. (See also CALL).

	Operation:	
(i)	$PC \leftarrow PC + k + 1$	Devices with 16 bits PC, 128K bytes program memory maximum.
(ii)	$PC \leftarrow PC + k + 1$	Devices with 22 bits PC, 8M bytes program memory maximum.

	Syntax:	**Operands:**	**Program Counter:**	**Stack**
(i)	RCALL k	$-2K \le k \le 2K$	$PC \leftarrow PC + k + 1$	$STACK \leftarrow PC+1$ $SP \leftarrow SP-2$ (2 bytes, 16 bits)
(ii)	RCALL k	$-2K \le k \le 2K$	$PC \leftarrow PC + k + 1$	$STACK \leftarrow PC+1$ $SP \leftarrow SP-3$ (3 bytes, 22 bits)

16 bit Opcode:

1101	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
         rcall   routine     ; Call subroutine
         ...
routine: push    r14         ; Save r14 on the stack
         ...
         pop     r14         ; Restore r14
         ret                 ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 3

RET - Return from Subroutine

Description:

Returns from subroutine. The return address is loaded from the STACK.

Operation:

(i) PC(15-0) ← STACK Devices with 16 bits PC, 128K bytes program memory maximum.
(ii) PC(21-0) ← STACK Devices with 22 bits PC, 8M bytes program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack
(i)	RET	None	See Operation	SP←SP+2,(2 bytes,16 bits pulled)
(ii)	RET	None	See Operation	SP←SP+3,(3 bytes,22 bits pulled)

16 bit Opcode:

1001	0101	0XX0	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
            call    routine     ; Call subroutine
            ...
routine:    push    r14         ; Save r14 on the stack
            ...
            pop     r14         ; Restore r14
            ret                 ; Return from subroutine
```

Words: 1 (2 bytes)
Cycles: 4

RETI - Return from Interrupt

Description:

Returns from interrupt. The return address is loaded from the STACK and the global interrupt flag is set.

Operation:

(i) PC(15-0) ← STACK Devices with 16 bits PC, 128K bytes program memory maximum.

(ii) PC(21-0) ← STACK Devices with 22 bits PC, 8M bytes program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack
(i)	RETI	None	See Operation	SP ← SP +2 (2 bytes, 16 bits)
(ii)	RETI	None	See Operation	SP ← SP +3 (3 bytes, 22 bits)

16 bit Opcode:

1001	0101	0XX1	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I: 1
The I flag is set.

Example:

```
            ...
extint:     push   r0        ; Save r0 on the stack
            ...
            pop    r0        ; Restore r0
            reti             ; Return and enable interrupts
```

Words: 1 (2 bytes)

Cycles: 4

6

RJMP - Relative Jump

Description:

Relative jump to an address within PC-2K and PC + 2K (words). In the assembler, labels are used instead of relative operands. For AVR microcontrollers with program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory from every address location.

Operation:

(i) $PC \leftarrow PC + k + 1$

	Syntax:	Operands:	Program Counter:	Stack
(i)	RJMP k	$-2K \le k \le 2K$	$PC \leftarrow PC + k + 1$	Unchanged

16 bit Opcode:

1100	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        cpi     r16,$42     ; Compare r16 to $42
        brne    error       ; Branch if r16 <> $42
        rjmp    ok          ; Unconditional branch
error:  add     r16,r17     ; Add r17 to r16
        inc     r16         ; Increment r16
ok:     nop                 ; Destination for rjmp (do nothing)
```

Words: 1 (2 bytes)
Cycles: 2

ROL - Rotate Left trough Carry

Description:

Shifts all bits in Rd one place to the left. The C flag is shifted into bit 0 of Rd. Bit 7 is shifted into the C flag.

Operation:

$$\boxed{C} \leftarrow \boxed{b7 \text{ - - - - - - - - - - - - - - - - - - } b0} \leftarrow \boxed{C}$$

	Syntax:	Operands:	Program Counter:
(i)	ROL Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode: (see ADC Rd,Rd)

0001	11dd	dddd	dddd

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: Rd3

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); Cleared otherwise (for values of N and C after the shift).

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: Rd7
Set if, before the shift, the MSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
          rolr15         ; Rotate left
          brcsoneenc     ; Branch if carry set
          ...
oneenc:   nop            ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

ROR - Rotate Right trough Carry

Description:

Shifts all bits in Rd one place to the right. The C flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C flag.

Operation:

$$\boxed{C} \rightarrow \boxed{b7 \text{ - - - - - - - - - - - - - - - - - - } b0} \rightarrow \boxed{C}$$

	Syntax:	Operands:	Program Counter:
(i)	ROR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	010d	dddd	0111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S: $N \oplus V$, For signed tests.

V: $N \oplus C$ (For N and C after the shift)
Set if (N is set and C is clear) or (N is clear and C is set); Cleared otherwise (for values of N and C after the shift).

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: Rd0
Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
            rorr15          ; Rotate right
            brcczeroenc     ; Branch if carry cleared
            ...
zeroenc:    nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1

SBC - Subtract with Carry

Description:

Subtracts two registers and subtracts with the C flag and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd - Rr - C$

	Syntax:	Operands:	Program Counter:
(i)	SBC Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

0000	10rd	dddd	rrrr

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet Rr3 + Rr3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{Rr7} \bullet \overline{R7} + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0} \bullet Z$
Previous value remains unchanged when the result is zero; cleared otherwise.

C: $\overline{Rd7} \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the contents of Rr plus previous carry is larger than the absolute value of the Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
                        ; Subtract r1:r0 from r3:r2
sub     r2,r0           ; Subtract low byte
sbc     r3,r1           ; Subtract with carry high byte
```

Words: 1 (2 bytes)
Cycles: 1

SBCI - Subtract Immediate with Carry

Description:

Subtracts a constant from a register and subtracts with the C flag and places the result in the destination register Rd.

	Operation:		
(i)	$Rd \leftarrow Rd - K - C$		
	Syntax:	**Operands:**	**Program Counter:**
(i)	SBCI Rd,K	$16 \le d \le 31,\ 0 \le K \le 255$	$PC \leftarrow PC + 1$

16 bit Opcode:

0100	KKKK	dddd	KKKK

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet K3 + K3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{K7} \bullet \overline{R7} + \overline{Rd7} \bullet K7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0} \bullet Z$
Previous value remains unchanged when the result is zero; cleared otherwise.

C: $\overline{Rd7} \bullet K7 + K7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the constant plus previous carry is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
                    ; Subtract $4F23 from r17:r16
    subi r16,$23    ; Subtract low byte
    sbci r17,$4F    ; Subtract with carry high byte
```

Words: 1 (2 bytes)
Cycles: 1

SBI - Set Bit in I/O Register

Description:

Sets a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers - addresses 0-31.

Operation:

(i) I/O(P,b) ← 1

	Syntax:	Operands:	Program Counter:
(i)	SBI P,b	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1

16 bit Opcode:

1001	1010	pppp	pbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
out  $1E,r0     ; Write EEPROM address
sbi  $1C,0      ; Set read bit in EECR
in   r1,$1D     ; Read EEPROM data
```

Words: 1 (2 bytes)

Cycles: 2

SBIC - Skip if Bit in I/O Register is Cleared

Description:

This instruction tests a single bit in an I/O register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O registers - addresses 0-31.

Operation:

(i) If I/O(P,b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBIC P,b	$0 \le P \le 31$, $0 \le b \le 7$	PC ← PC + 1, If condition is false, no skip. PC ← PC + 2, If next instruction is one word. PC ← PC + 3, If next instruction is JMP or CALL

16 bit Opcode:

1001	1001	pppp	pbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
e2wait: sbic  $1C,1      ; Skip next inst. if EEWE cleared
        rjmp  e2wait     ; EEPROM write not finished
        nop              ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

SBIS - Skip if Bit in I/O Register is Set

Description:
This instruction tests a single bit in an I/O register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O registers - addresses 0-31.

Operation:

(i) If I/O(P,b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBIS P,b	$0 \le P \le 31, 0 \le b \le 7$	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a JMP or a CALL

16 bit Opcode:

1001	1011	pppp	pbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
waitset: sbis  $10,0        ; Skip next inst. if bit 0 in Port D set
         rjmp  waitset      ; Bit not set
         nop                ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

SBIW - Subtract Immediate from Word

Description:

Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the pointer registers.

Operation:

(i) Rdh:Rdl ← Rdh:Rdl - K

	Syntax:	Operands:	Program Counter:
(i)	SBIW Rdl,K	dl ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16 bit Opcode:

1001	0111	KKdd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	⇔	⇔	⇔	⇔

S: $N \oplus V$, For signed tests.

V: $Rdh7 \bullet \overline{R15}$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R15$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$0000; cleared otherwise.

C: $R15 \bullet \overline{Rdh7}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
sbiw   r24,1        ; Subtract 1 from r25:r24
sbiw   r28,63       ; Subtract 63 from the Y pointer(r29:r28)
```

Words: 1 (2 bytes)
Cycles: 2

SBR - Set Bits in Register

Description:

Sets specified bits in register Rd. Performs the logical ORI between the contents of register Rd and a constant mask K and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd v K

	Syntax:	Operands:	Program Counter:
(i)	SBR Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	PC ← PC + 1

16 bit Opcode:

0110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
sbr   r16,3         ; Set bits 0 and 1 in r16
sbr   r17,$F0       ; Set 4 MSB in r17
```

Words: 1 (2 bytes)
Cycles: 1

SBRC - Skip if Bit in Register is Cleared

Description:

This instruction tests a single bit in a register and skips the next instruction if the bit is cleared.

Operation:

(i) If Rr(b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBRC Rr,b	$0 \le r \le 31$, $0 \le b \le 7$	PC ← PC + 1, If condition is false, no skip. PC ← PC + 2, If next instruction is one word. PC ← PC + 3, If next instruction is JMP or CALL

16 bit Opcode:

1111	110r	rrrr	Xbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
sub   r0,r1      ; Subtract r1 from r0
sbrc  r0,7       ; Skip if bit 7 in r0 cleared
sub   r0,r1      ; Only executed if bit 7 in r0 not cleared
nop              ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

SBRS - Skip if Bit in Register is Set

Description:

This instruction tests a single bit in a register and skips the next instruction if the bit is set.

Operation:

(i) If Rr(b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBRS Rr,b	$0 \le r \le 31$, $0 \le b \le 7$	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a JMP or a CALL

16 bit Opcode:

1111	111r	rrrr	Xbbb

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
sub   r0,r1   ; Subtract r1 from r0
sbrs  r0,7    ; Skip if bit 7 in r0 set
neg   r0      ; Only executed if bit 7 in r0 not set
nop           ; Continue (do nothing)
```

Words: 1 (2 bytes)

Cycles: 1 if condition is false (no skip)
2 if condition is true (skip is executed)

SEC - Set Carry Flag

Description:

Sets the Carry flag (C) in SREG (status register).

Operation:

(i) C ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEC	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0000	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	1

C: 1
Carry flag set

Example:

```
sec             ; Set carry flag
adc     r0,r1   ; r0=r0+r1+1
```

Words: 1 (2 bytes)

Cycles: 1

SEH - Set Half Carry Flag

Description:

Sets the Half Carry (H) in SREG (status register).

Operation:

(i) H ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEH	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0101	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	1	-	-	-	-	-

H: 1
Half Carry flag set

Example:

```
        seh          ; Set Half Carry flag
```

Words: 1 (2 bytes)
Cycles: 1

6

SEI - Set Global Interrupt Flag

Description:

Sets the Global Interrupt flag (I) in SREG (status register).

Operation:

(i) $I \leftarrow 1$

	Syntax:	Operands:	Program Counter:
(i)	SEI	None	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	0100	0111	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
1	-	-	-	-	-	-	-

I: 1
Global Interrupt flag set

Example:

```
cli                 ; Disable interrupts
in    r13,$16       ; Read Port B
sei                 ; Enable interrupts
```

Words: 1 (2 bytes)
Cycles: 1

SEN - Set Negative Flag

Description:

Sets the Negative flag (N) in SREG (status register).

	Operation:
(i)	N ← 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	SEN	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0010	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	1	-	-

N: 1
Negative flag set

Example:

```
add   r2,r19        ; Add r19 to r2
sen                 ; Set negative flag
```

Words: 1 (2 bytes)
Cycles: 1

SER - Set all bits in Register

Description:

Loads $FF directly to register Rd.

	Operation:
(i)	Rd ← $FF

	Syntax:	Operands:	Program Counter:
(i)	SER Rd	$16 \leq d \leq 31$	PC ← PC + 1

16 bit Opcode:

1110	1111	dddd	1111

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr  r16          ; Clear r16
ser  r17          ; Set r17
out  $18,r16      ; Write zeros to Port B
nop               ; Delay (do nothing)
out  $18,r17      ; Write ones to Port B
```

Words: 1 (2 bytes)

Cycles: 1

SES - Set Signed Flag

Description:

Sets the Signed flag (S) in SREG (status register).

Operation:

(i) S ← 1

	Syntax:	Operands:	Program Counter:
(i)	SES	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0100	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	1	-	-	-	-

S: 1
Signed flag set

Example:

```
add   r2,r19          ; Add r19 to r2
ses                   ; Set negative flag
```

Words: 1 (2 bytes)
Cycles: 1

SET - Set T Flag

Description:

Sets the T flag in SREG (status register).

	Operation:
(i)	T ← 1

	Syntax:	**Operands:**	**Program Counter:**
(i)	SET	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0110	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	1	-	-	-	-	-	-

T: 1
T flag set

Example:

```
set                    ; Set T flag
```

Words: 1 (2 bytes)
Cycles: 1

SEV - Set Overflow Flag

Description:
Sets the Overflow flag (V) in SREG (status register).

Operation:
(i) V ← 1

	Syntax:	Operands:	Program Counter:
(i)	SEV	None	PC ← PC + 1

16 bit Opcode:

1001	0100	0011	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	1	-	-	-

V: 1
Overflow flag set

Example:

```
add  r2,r19        ; Add r19 to r2
sev                ; Set overflow flag
```

Words: 1 (2 bytes)
Cycles: 1

6

SEZ - Set Zero Flag

Description:

Sets the Zero flag (Z) in SREG (status register).

Operation:

(i) $Z \leftarrow 1$

	Syntax:	Operands:	Program Counter:
(i)	SEZ	None	$PC \leftarrow PC + 1$

16 bit Opcode:

1001	0100	0001	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	1	-

Z: 1
Zero flag set

Example:

```
add  r2,r19        ; Add r19 to r2
sez                ; Set zero flag
```

Words: 1 (2 bytes)
Cycles: 1

SLEEP

Description:

This instruction sets the circuit in sleep mode defined by the MCU control register. When an interrupt wakes up the MCU from a sleep state, the instruction following the SLEEP instruction will be executed before the interrupt handler is executed.

Operation:

Syntax:	**Operands:**	**Program Counter:**
SLEEP	None	PC ← PC + 1

16 bit Opcode:

1001	0101	100X	1000

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
        mov     r0,r11          ; Copy r11 to r0
        sleep                   ; Put MCU in sleep mode
```

Words: 1 (2 bytes)
Cycles: 1

ST - Store Indirect From Register to SRAM using Index X

Description:

Stores one byte indirect from Register to SRAM. The SRAM location is pointed to by the X (16 bits) pointer register in the register file. Memory access is limited to the current SRAM Page of 64K bytes. To access another SRAM page the RAMPX register in the I/O area has to be changed.

The X pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the X pointer register.

Using the X pointer:

	Operation:		Comment:
(i)	$(X) \leftarrow Rr$		X: Unchanged
(ii)	$(X) \leftarrow Rr$	$X \leftarrow X+1$	X: Post incremented
(iii)	$X \leftarrow X - 1$	$(X) \leftarrow Rr$	X: Pre decremented

	Syntax:	Operands:	Program Counter:
(i)	ST X, Rr	$0 \le r \le 31$	$PC \leftarrow PC + 1$
(ii)	ST X+, Rr	$0 \le r \le 31$	$PC \leftarrow PC + 1$
(iii)	ST -X, Rr	$0 \le r \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode :

(i)	1001	001r	rrrr	1100
(ii)	1001	001r	rrrr	1101
(iii)	1001	001r	rrrr	1110

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr   r27        ; Clear X high byte
ldi   r26,$20    ; Set X low byte to $20
st    X+,r0      ; Store r0 in SRAM loc. $20(X post inc)
st    X,r1       ; Store r1 in SRAM loc. $21
ldi   r26,$23    ; Set X low byte to $23
st    r2,X       ; Store r2 in SRAM loc. $23
st    r3,-X      ; Store r3 in SRAM loc. $22(X pre dec)
```

Words: 1 (2 bytes)
Cycles: 2

ST (STD) - Store Indirect From Register to SRAM using Index Y

Description:

Stores one byte indirect with or without displacement from Register to SRAM. The SRAM location is pointed to by the Y (16 bits) pointer register in the register file. Memory access is limited to the current SRAM Page of 64K bytes. To access another SRAM page the RAMPY register in the I/O area has to be changed.

The Y pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are especially suited for stack pointer usage of the Y pointer register.

Using the Y pointer:

	Operation:		Comment:
(i)	(Y) ← Rr		Y: Unchanged
(ii)	(Y) ← Rr	Y ← Y+1	Y: Post incremented
(iii)	Y ← Y - 1	(Y) ← Rr	Y: Pre decremented
(iiii)	(Y+q) ← Rr		Y: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Y+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Y, Rr	$0 \le r \le 31$	PC ← PC + 1
(iiii)	STD Y+q, Rr	$0 \le r \le 31$, $0 \le q \le 63$	PC ← PC + 1

16 bit Opcode :

(i)	1000	001r	rrrr	1000
(ii)	1001	001r	rrrr	1001
(iii)	1001	001r	rrrr	1010
(iiii)	10q0	qq1r	rrrr	1qqq

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr   r29        ; Clear Y high byte
ldi   r28,$20    ; Set Y low byte to $20
st    Y+,r0      ; Store r0 in SRAM loc. $20(Y post inc)
st    Y,r1       ; Store r1 in SRAM loc. $21
ldi   r28,$23    ; Set Y low byte to $23
st    Y,r2       ; Store r2 in SRAM loc. $23
st    -Y,r3      ; Store r3 in SRAM loc. $22(Y pre dec)
std   Y+2,r4     ; Store r4 in SRAM loc. $24
```

Words: 1 (2 bytes)
Cycles: 2

ST (STD) - Store Indirect From Register to SRAM using Index Z

Description:

Stores one byte indirect with or without displacement from Register to SRAM. The SRAM location is pointed to by the Z (16 bits) pointer register in the register file. Memory access is limited to the current SRAM Page of 64K bytes. To access another SRAM page the RAMPZ register in the I/O area has to be changed.

The Z pointer register can either be left unchanged after the operation, or it can be incremented or decremented. These features are very suited for stack pointer usage of the Z pointer register, but because the Z pointer register can be used for indirect subroutine calls, indirect jumps and table lookup it is often more convenient to use the X or Y pointer as a dedicated stack pointer.

Using the Z pointer:

	Operation:		Comment:
(i)	(Z) ←Rr		Z: Unchanged
(ii)	(Z) ← Rr	Z ← Z+1	Z: Post incremented
(iii)	Z ← Z - 1	(Z) ← Rr	Z: Pre decremented
(iiii)	(Z+q) ← Rr		Z: Unchanged, q: Displacement

	Syntax:	Operands:	Program Counter:
(i)	ST Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(ii)	ST Z+, Rr	$0 \le r \le 31$	PC ← PC + 1
(iii)	ST -Z, Rr	$0 \le r \le 31$	PC ← PC + 1
(iiii)	STD Z+q, Rr	$0 \le r \le 31, 0 \le q \le 63$	PC ← PC + 1

16 bit Opcode :

(i)	1000	001r	rrrr	0000
(ii)	1001	001r	rrrr	0001
(iii)	1001	001r	rrrr	0010
(iiii)	10q0	qq1r	rrrr	0qqq

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
clr     r31         ; Clear Z high byte
ldi     r30,$20     ; Set Z low byte to $20
st      Z+,r0       ; Store r0 in SRAM loc. $20(Z post inc)
st      Z,r1        ; Store r1 in SRAM loc. $21
ldi     r30,$23     ; Set Z low byte to $23
st      Z,r2        ; Store r2 in SRAM loc. $23
st      -Z,r3       ; Store r3 in SRAM loc. $22(Z pre dec)
std     Z+2,r4      ; Store r4 in SRAM loc. $24
```

Words: 1 (2 bytes)
Cycles: 2

STS - Store Direct to SRAM

Description:

Stores one byte from a Register to the SRAM. A 16-bit address must be supplied. Memory access is limited to the current SRAM Page of 64K bytes. The SDS instruction uses the RAMPZ register to access memory above 64K bytes.

Operation:

(i) $(k) \leftarrow Rr$

	Syntax:	Operands:	Program Counter:
(i)	STS k,Rr	$0 \le r \le 31$, $0 \le k \le 65535$	$PC \leftarrow PC + 2$

32 bit Opcode:

1001	001d	dddd	0000
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
lds     r2,$FF00        ; Load r2 with the contents of SRAM location $FF00
add     r2,r1           ; add r1 to r2
sts     $FF00,r2        ; Write back
```

Words: 2 (4 bytes)

Cycles: 3

SUB - Subtract without Carry

Description:

Subtracts two registers and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd - Rr

	Syntax:	Operands:	Program Counter:
(i)	SUB Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16 bit Opcode:

0001	10rd	dddd	rrrr

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet Rr3 + Rr3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{Rr7} \bullet \overline{R7} + \overline{Rd7} \bullet Rr7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $\overline{Rd7} \bullet Rr7 + Rr7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of the contents of Rr is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        sub    r13,r12      ; Subtract r12 from r13
        brne   noteq        ; Branch if r12<>r13
        ...
noteq:  nop                 ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

SUBI - Subtract Immediate

Description:

Subtracts a register and a constant and places the result in the destination register Rd. This instruction is working on Register R16 to R31 and is very well suited for operations on the X, Y and Z pointers.

Operation:

(i) $Rd \leftarrow Rd - K$

	Syntax:	Operands:	Program Counter:
(i)	SUBI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	$PC \leftarrow PC + 1$

16 bit Opcode:

0101	KKKK	dddd	KKKK

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	⇔	⇔	⇔	⇔	⇔	⇔

H: $\overline{Rd3} \bullet K3 + K3 \bullet R3 + R3 \bullet \overline{Rd3}$
Set if there was a borrow from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet \overline{K7} \bullet \overline{R7} + \overline{Rd7} \bullet K7 \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: $R7$
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

C: $\overline{Rd7} \bullet K7 + K7 \bullet R7 + R7 \bullet \overline{Rd7}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
        subir22,$11     ; Subtract $11 from r22
        brnenoteq       ; Branch if r22<>$11
        ...
noteq:  nop             ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

SWAP - Swap Nibbles

Description:

Swaps high and low nibbles in a register.

Operation:

(i) R(7-4) ← Rd(3-0), R(3-0) ← Rd(7-4)

	Syntax:	Operands:	Program Counter:
(i)	SWAP Rd	$0 \le d \le 31$	PC ← PC + 1

16 bit Opcode:

1001	010d	dddd	0010

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

R (Result) equals Rd after the operation.

Example:

```
inc   r1        ; Increment r1
swap  r1        ; Swap high and low nibble of r1
inc   r1        ; Increment high nibble of r1
swap  r1        ; Swap back
```

Words: 1 (2 bytes)
Cycles: 1

TST - Test for Zero or Minus

Description:

Tests if a register is zero or negative. Performs a logical AND between a register and itself. The register will remain unchanged.

Operation:

(i) $Rd \leftarrow Rd \cdot Rd$

	Syntax:	Operands:	Program Counter:
(i)	TST Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16 bit Opcode:

0010	00dd	dddd	dddd

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	⇔	0	⇔	⇔	-

S: $N \oplus V$, For signed tests.

V: 0
Cleared

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$00; cleared otherwise.

R (Result) equals Rd.

Example:

```
        tst   r0          ; Test r0
        breq  zero        ; Branch if r0=0
        ...
zero:   nop               ; Branch destination (do nothing)
```

Words: 1 (2 bytes)
Cycles: 1

WDR - Watchdog Reset

Description:

This instruction resets the Watchdog Timer. This instruction must be executed within a limited time given by the WD prescaler. See the Watchdog Timer hardware specification.

Operation:

(i) WD timer restart.

	Syntax:	Operands:	Program Counter:
(i)	WDR	None	PC ← PC + 1

16 bit Opcode:

1001	0101	101X	1000

Status Register and Boolean Formulae:

I	T	H	S	V	N	Z	C
-	-	-	-	-	-	-	-

Example:

```
wdr                ; Reset watchdog timer
```

Words: 1 (2 bytes)
Cycles: 1

AVR Development Tools

This section describes some of the development tools that are available for the 8-bit *AVR* family.

- **Atmel AVR Assembler**
- **Atmel AVR Simulator**
- **IAR ANSI C-Compiler, Assembler, Linker, Librarian & Debugger**
- **Atmel In-Circuit Emulator (ICE)**

There are a lot of development tools under development, please contact Atmel for more details.

8-Bit AVR® Development Tools

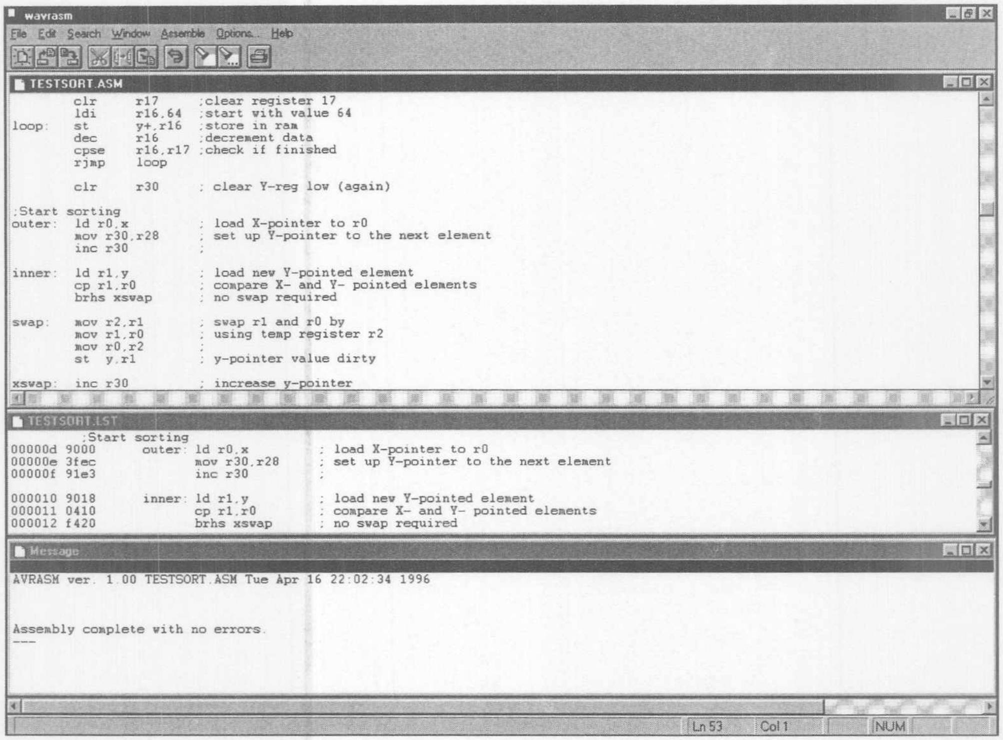
7

0857A

Atmel *AVR* Assembler

Features:

- Translates Assembler source programs into object code
- Extremely fast assembling
- Supports all the microcontrollers in the AT90S family
- Powerful Macro capabilities
- Supports all standard output formats
- Easy to use MS-Windows interface
- Editor included in MS-Windows version
- Jump to next/previous error
- Also available in MS-DOS command line version



Powerful Macro Capabilities

The Assembler contains powerful macro capabilities, enabling the user to build a virtual instruction set which is structures of ordinary *AVR* instructions. For example, this macro does a 16 bit subtraction:

```
;
; SUB16 macro definition
```

```
; The macro subtracts a 16 bit constant from a register
; pair. A call to the macro is done by
; SUB16 Regh,Regl,Const
;
.MACRO SUB16              ; Macro name
  subi$1,low($2)          ; subtract low byte
  sbci $0,high($2)        ; subtract high byte
.ENDMACRO

; ...

; Call the macro
  ldi   r16,low(0x3400)   ; set values in registers
  ldi   r17,low(0x3400)   ;
  SUB16 r17,r16,0x23a0    ; compute 0x3400-0x23a0
```

Assembly Directives

The assembler supports a number of directives making the application development easier. In addition to the directives for macro generation and control, the assembler contains directives for:

- Including files. Included files can be nested.
- Set program origin.
- Symbol usage. The user can define symbols and labels and refer to these throughout the assembly program.
- Constant data initialization. The user can do constant initialization. Constants will be placed in the Flash program memory.
- List file control.
- Support of expressions in a C-like syntax.

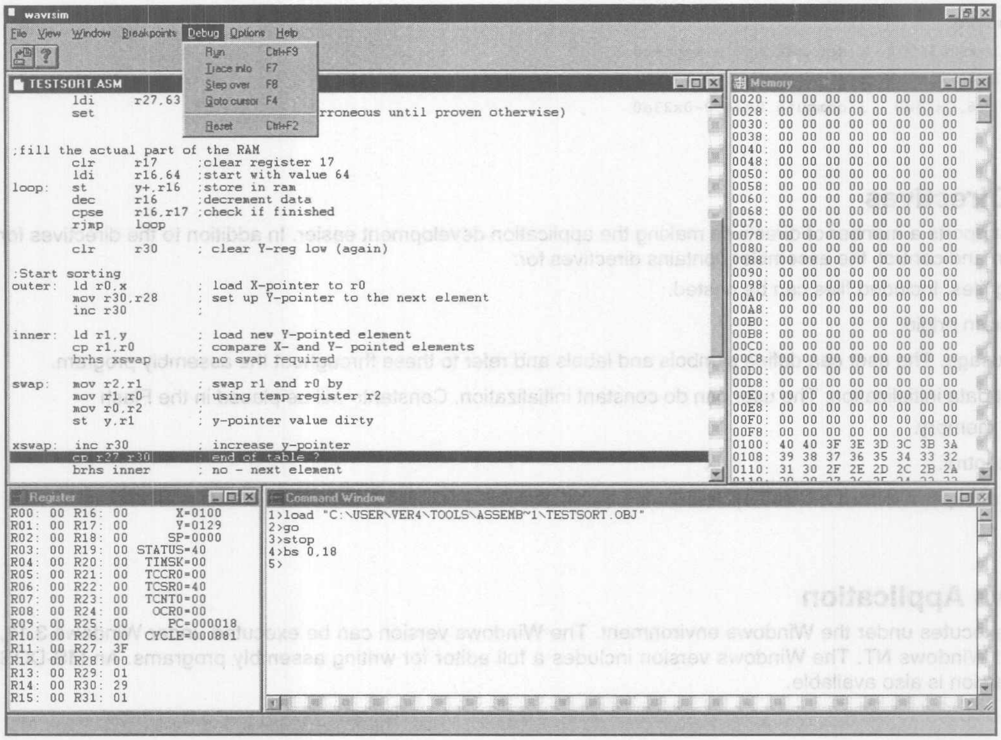
MS Windows Application

The assembler executes under the Windows environment. The Windows version can be executed under Windows 3.11, Windows 95 and Windows NT. The Windows version includes a full editor for writing assembly programs. An MS-DOS command line version is also available.

Atmel *AVR* Simulator

Features:

- Supports the whole range of AT90S microcontrollers
- Assembly source level simulation
- Powerful debugging facilities
- Full support of *AVR* peripheral devices
- Easy to use MS-Windows interface



Debugging Facilities

The simulator has a number of functions to help the programmer to debug programs including:

- Breakpoints: Set up to 256 breakpoints in the source window, and program execution will halt upon reaching one of the breakpoints.
- Single stepping: Step through the code instruction by instruction and watch the execution.
- Step into/Trace over: Select whether calls should be traced, or if these simulation details should be omitted.
- Goto cursor: Place the cursor on an instruction, and the simulator will execute until the marked instruction is reached.
- Run from file. The user can write scripts consisting of simulator commands.

- Display of registers and memory. The user can view all memory spaces, all general purpose working registers and the registers in the I/O map. The user also has the ability to write values in these memories and registers.
- Logging. All information, including register contents, memory contents and I/O accesses can be logged for each instruction.
- The simulator holds control on the number of clock cycles elapsed.

All commands are available through a command window and through menus.

Simulation of Peripherals

- The simulator supports the peripheral devices of the *AVR* microcontrollers, including:
- Interrupts. Each interrupt can be set in each cycle enabling complex combinations including nested interrupts.
- Timer/Counters. The timer/counters can also be simulated. This of course includes generating interrupts on overflows and compare matches. Free running mode is also supported.
- I/O ports. The I/O ports are implemented, giving the user the ability to communicate with the simulated programs through the ports.

Together with the powerful control mechanisms present, this makes the simulator a complete debugging tool for the *AVR* family of Enhanced RISC Microcontrollers.

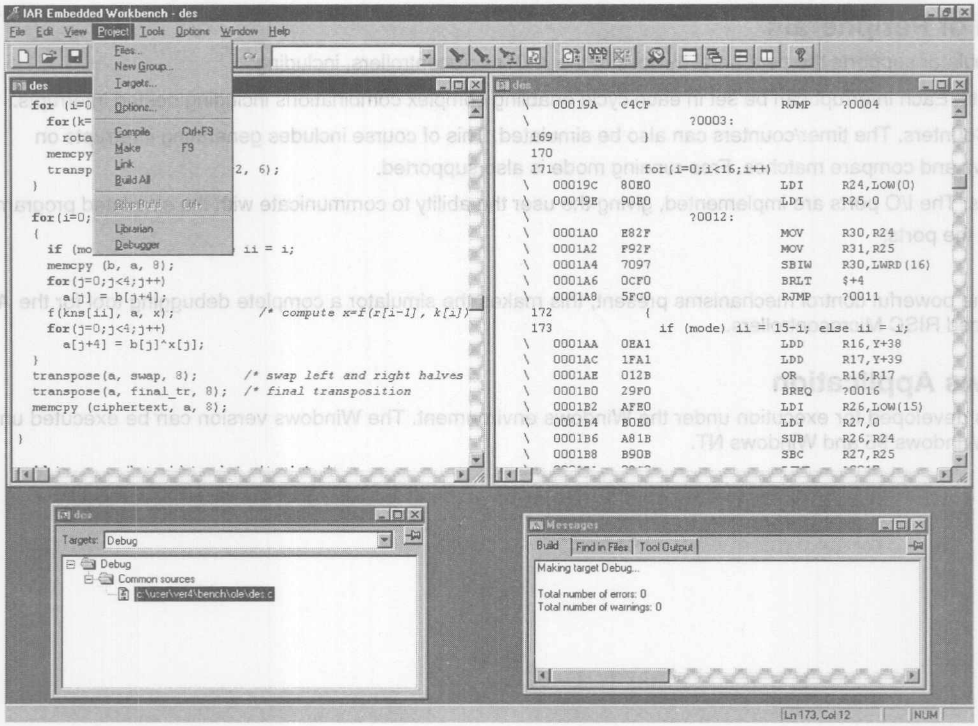
MS Windows Application

The simulator is developed for execution under the Windows environment. The Windows version can be executed under Windows 3.11, Windows 95 and Windows NT.

IAR Development Tools for Atmel *AVR*

Features:

- Fully ANSI Compatible C Compiler
- Includes Embedded Workbench
- C and Assembler level language debugger
- Runs under DOS, Windows 3.11, Windows 95 and Windows NT



Embedded Workbench

The Embedded Workbench offers a total integration of C compiler, editor, linker, librarian, assembler, editor and debugger in a seamless environment. The Embedded Workbench includes the following features:

- Flexible editor. The editor offers flexibility in terms of customizable toolbar and user defined shortcuts. The editor implements the basic Windows editing commands as well as extensions for C programming, such as C syntax coloring and direct jump to context from error listing.
- The hierarchical project maintenance facility makes it possible to have several targets, such as a release target and a debug target with different option settings. Each target is built from one or several groups which in turn are built from one or several files. Compiler options can be set on each level.

- The Make system automatically generates a dependency list of output files, source files and include files. This allows the Make system to only compile, recompile or reassemble the updated parts of the source code, which speeds up the building process.
- Extensive on-line help function makes it easy to quickly find specific help about the tool without leaving the Embedded Workbench.
- The compiler is also available under MS-DOS in a mouse controlled menu driven user interface, allowing all development steps to be performed in an integrated DOS environment.

C Compiler

The C Compiler is an optimizing ANSI C compatible C compiler. The C compiler is the core product in the Embedded Workbench. All data types required by ANSI are supported. Full ANSI C compatibility also means that the compiler conforms to all requirements placed by ANSI on run-time behavior. The C Compiler includes the following features:

- Fully compatible with the ANSI C standard
- Fully reentrant code
- Absolute read/write of I/O locations at C level
- Built-in AT90S specific Optimizer
- AT90S specific extensions
- Full floating point support
- Four different memory models to allow best fit selection

Assembler

The C compiler kit comes with a relocatable structured assembler. This provides the option of coding time critical sections of the application in assembly without loosing the advantages of the C language. The preprocessor of the C language is incorporated in the assembler, thus allowing use of the full ANSI C macro language, with conditional assembly, macro definitions, if statements and more. C include files can also be used in an assembly program. All modules written in assembly can easily be accessed from C and vice-versa, making the interface between C and assembly a straightforward process.

Linker

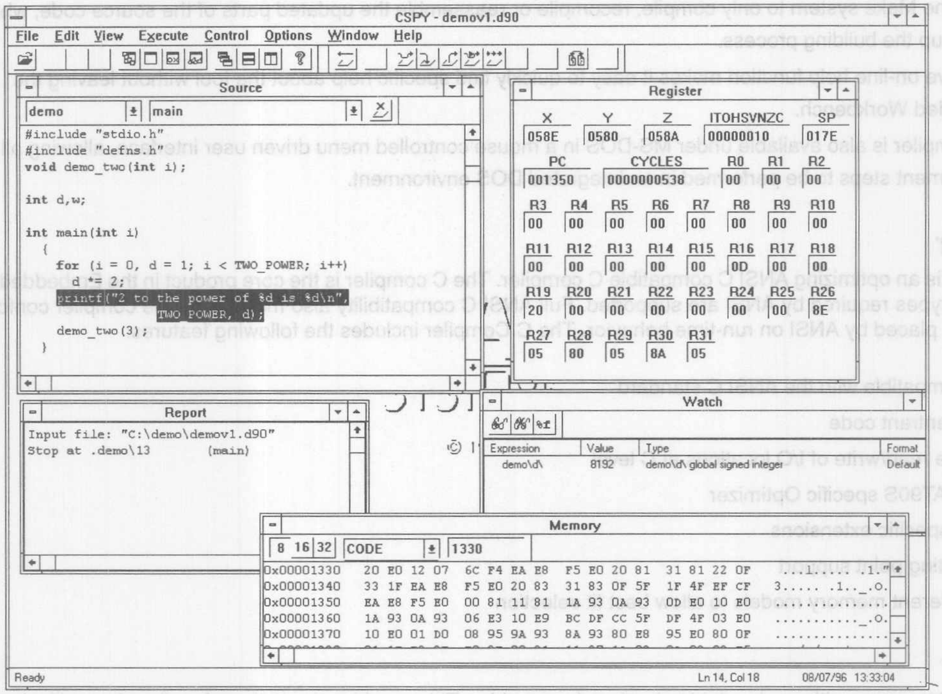
The linker XLINK supports complete linking, relocation and format generation to produce AT90S code. Over 30 output formats are supported. Detailed cross reference and map listing with segments, symbol information, variable locations and function addresses are easily generated.

Librarian

The librarian XLIB creates and maintains libraries and library modules. Listings of modules, entry points and symbolic information contained in every library are easily generated. XLIB can also give the library attribute for conditional loading or the program attribute for unconditional loading.

C-SPY High Level Language Debugger and Simulator

The C-SPY high level language debugger/simulator combines the detailed control code execution needed for embedded development debugging with the flexibility and power of the C language. The source window can display C source and mix it with assembly source. No extra hardware is needed since instruction execution is simulated.



C-SPY includes the following features:

- Powerful breakpoint setting. C-SPY has an unlimited number of breakpoints. Breakpoints can be set on C statements, assembler instructions, and on any address with access type of read, write and opcode fetch. It may also be set as a combination of these. After triggering a breakpoint, a macro command can be executed.
- C-like macro language. A powerful C-like macro language can tailor the environment used for debugging in the C-SPY, including system macros for host file I/O simulation, reset, start up and shut down, as well as statements such as for loops, while loops, if and return statements.
- Interrupt simulation implements commands to launch specific interrupts at a specific cycle count or periodically.
- I/O simulation. C-SPY terminal I/O emulation offers a console window for target system I/O. This unique feature is useful for debugging embedded applications when logical flows are of interest or the target is not yet ready.
- The watch points window makes it possible to watch any expression. The window itself will be updated whenever a breakpoint is gritted or a step is finished. Any variable can be modified during the execution by using specific C expressions.
- The source window for the assembler debugger displays the assembler instructions. It has a built-in assembler and disassembler function, menu and register window, and can evaluate assembler expressions.

Atmel *AVR* In-Circuit Emulator

Features:

- Supports the whole range of AT90S microcontrollers
- Full visibility of all MCU resources
- Powerful breakpoint facilities
- Extensive execution control
- 32K x 96 bit wide Trace Buffer for real-time data collection
- 32-bit Time Stamp generator
- 8-Bit Event memory for event generation
- Supports 3 download modes
- Real time emulation
- Software adjustable clock speed
- Supports assembler and C source level debugging
- Supports all on-chip peripherals
- Serial- and parallel port interfacing
- Includes simple programmer
- Integrated with other *AVR* development tools

Full visibility

Using the emulator, the status of all resources can be monitored, and most of them can also be modified:

- The register file (R/W)
- SRAM (R/W)
- Program memory (R/W)
- EEPROM (R/W)
- Program Counter (R/W)
- I/O locations (R/W)

7

Powerful breakpoint facilities

The emulator incorporates powerful breakpoint facilities including:

- SRAM address breakpoint: Break when a specified address in the SRAM is read or written.
- SRAM data breakpoint: Break when a specified value is written to or read from SRAM.
- SRAM address and data breakpoint: An SRAM address breakpoint can be combined with an SRAM data breakpoint.
- Program memory address breakpoint: Break when a specified program memory address is accessed.
- Program memory data breakpoint: Break when a specified value is read from the program memory.
- Register match breakpoint: Break when a specified value is read/written from/to one of the 32 registers.
- External trigger breakpoint: Break when an external signal is rising or falling.

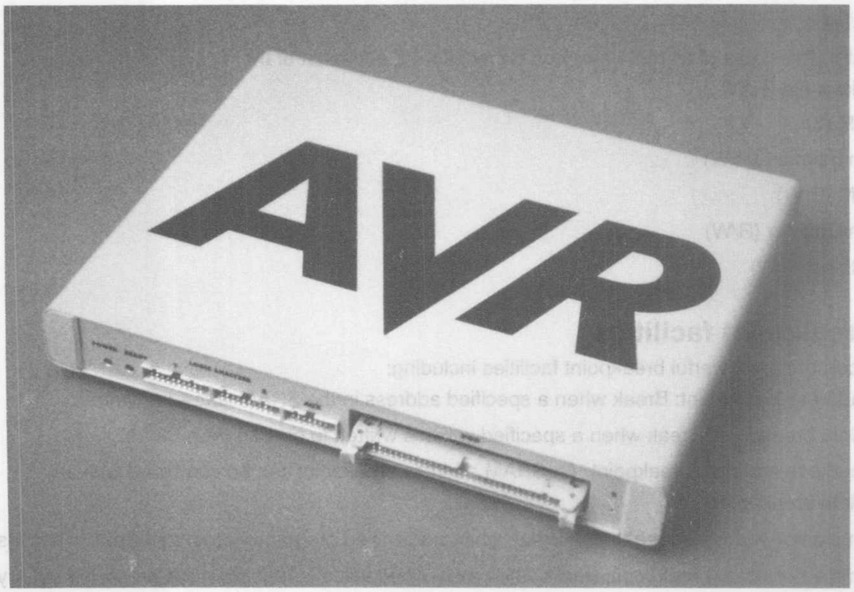
Extensive execution control

The emulator features extensive execution control:

- Single step execution: The emulator executes one instruction and then stops.
- Multiple step execution: The emulator executes a specified number of instructions and then stops.
- Software controlled Trace into/Step over.
- Start/Resume/Stop execution.
- Reset emulator.

Miscellaneous

- 5 trigger outputs are provided for connection to a DSO or a Logic Analyzer.
- Serial- and parallel port interface. The emulator can be connected to the PC through a standard serial- or parallel port.
- Simple programmer. A device present in the socket can be programmed from the PC or from the emulators program memory.
- In-circuit programming capabilities.
- Supports a wide range of download file formats like Intel Hex and Motorola S-Records.
- Fast download time.



Technical Specifications

System Unit

Physical Dimensions . . . . (H x W x D) 32.4 x 277.1 x 218.6 mm / 1.3 x 10.8 x 8.5 in

Weight . . . . 400g / 0.88 lbs

Power Voltage Requirements . . . . 9 - 15 VDC

Power Consumption . . . . < 20W

ICE Power Consumption 10W

Max. Application Power Consumption 5W

Ambient Temperature . . . . -40 - +85°C (Operating)
-55 - +85°C (Non-Operating)

Relative Humidity (Non-condensing) . . . . 10 - 90 % (Operating)
5 - 95 % (Non-Operating)

Shock . . . . 20 g, 11ms half sine

Vibration 5g

Connections

Power
Connector . . . . 5.5 mm OD/ 2.1mm ID Center Negative

Host
Serial Connector (RS-232) . . . . 9-pin D-SUB Female

Serial Communications Speed . . . . 9600 - 115,200 bits/s

Parallel Connector (LPT) . . . . 25-pin D-SUB Male

Parallel Communications Speed (Max) . . . . 80 kbyte/s

Pod
Emulating ≤ 40 pins . . . . one 2 x 32 Male Header

External Trigger Inputs / Outputs
Connector . . . . 2 x 7 Male Header

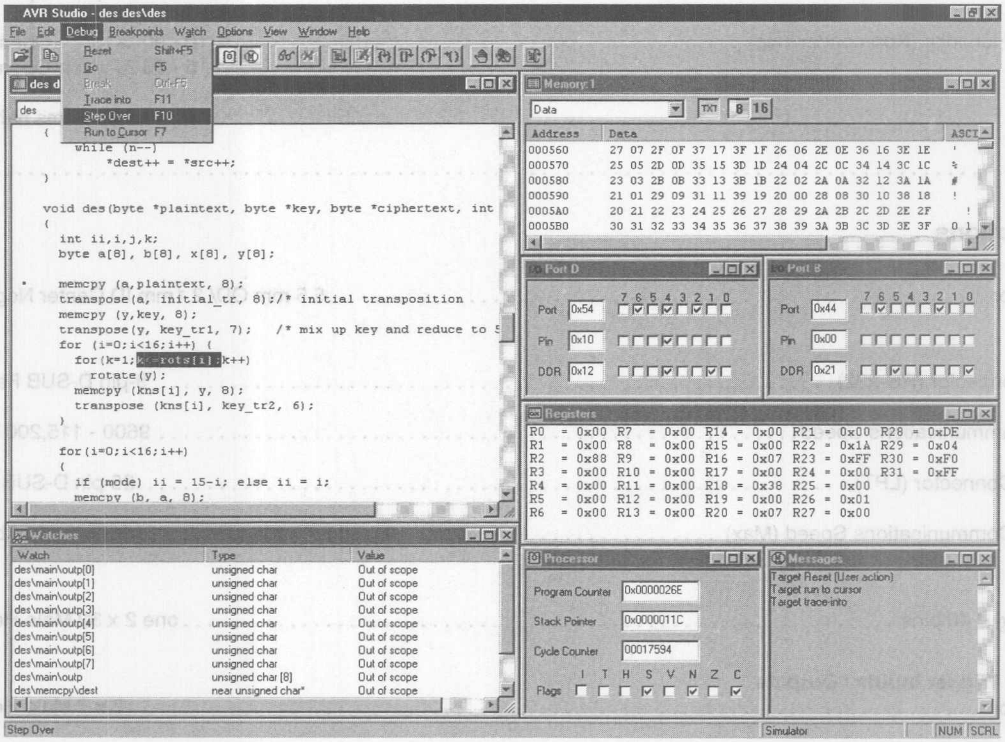
Logic Analyzer Interface
Connectors . . . . two 2 x 10 Male Headers

AVR Programmer Module Interface
Connector . . . . 2 x 5 Male Header

Atmel *AVR* Studio

Features:

- Supports the full range of AT90S microcontrollers
- Enables source level debugging of C and Assembly source code
- Allows for Assembly level debugging of C source code
- Interfaces the AVR In-Circuit Emulator for Real Time execution
- Built-in AVR instruction set Simulator allowing for debugging when no Emulator connected
- Watch for viewing and modifying symbols such as plain variables, structures and unions
- Full visibility of all MCU memories: Register file, SRAM, Program memory and EEPROM
- Exploits all features of the AVR In-Circuit Emulator
- Powerful breakpoint facilities
- Extensive execution control
- Supports IAR's ICCA90 C Compiler, IAR's AA90 Assembler and Atmel's AVR Assembler



Contents

8

Table of Contents

Each Atmel data sheet includes an Ordering Information Section which specifies the package types available. This section provides size specifications and outlines for all package types. [1]

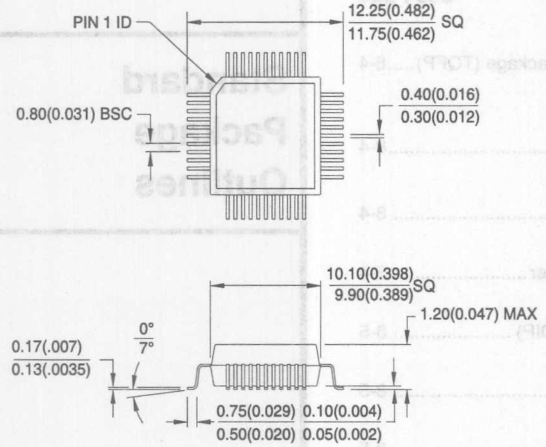
Standard Package Outlines

8

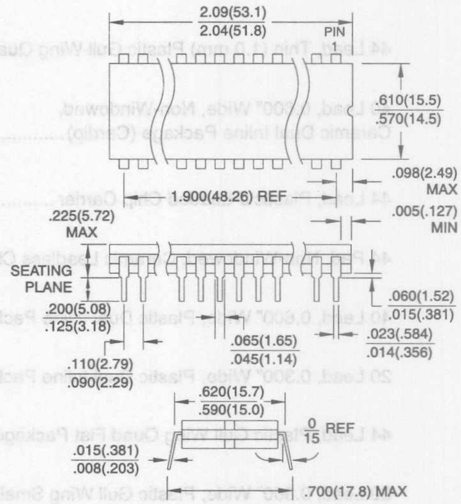
Note: 1. Dimensions shown do not include lead plating or mold flash.

0858A

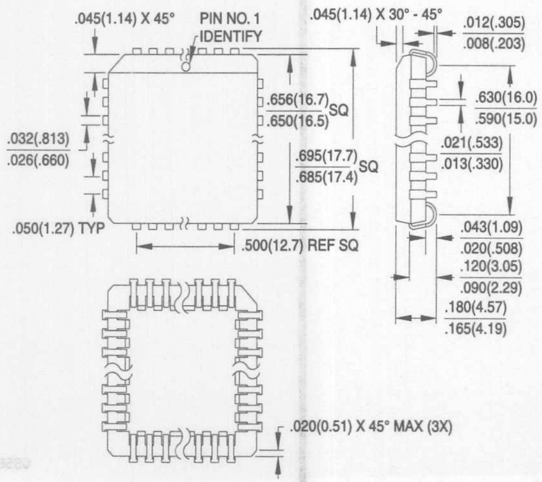
44A, 44 Lead, Thin (1.0 mm)
Plastic Gull Wing Quad Flat Package (TQFP)
Dimensions in Millimeters and (Inches)



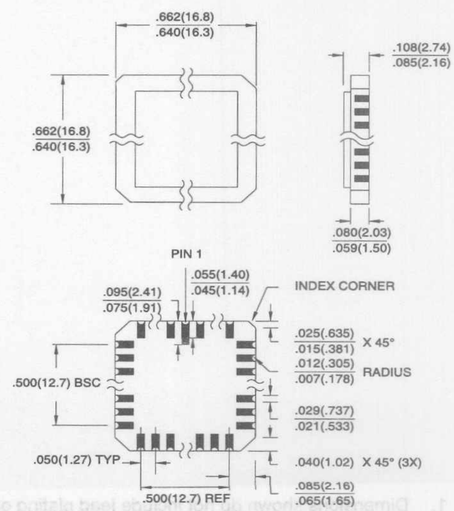
40D6, 40 Lead, 0.600" Wide, Non-Windowed,
Ceramic Dual Inline Package (Cerdip)
Dimensions in Inches and (Millimeters)
MIL-STD-1835 D-5 CONFIG A



44J, 44 Lead, Plastic J-Leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC

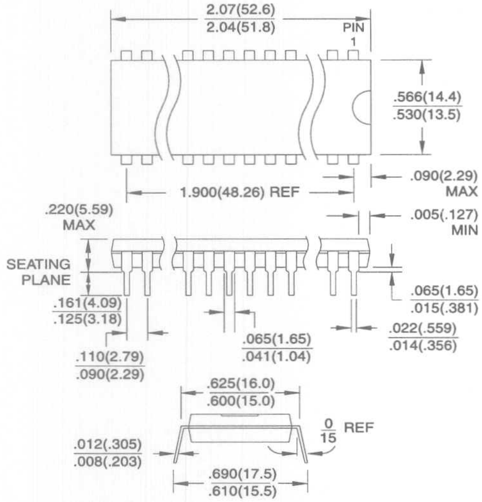


44L, 44 Lead, Non-Windowed,
Ceramic Leadless Chip Carrier (LCC)
Dimensions in Inches and (Millimeters)*
MIL-STD-1835 C-5

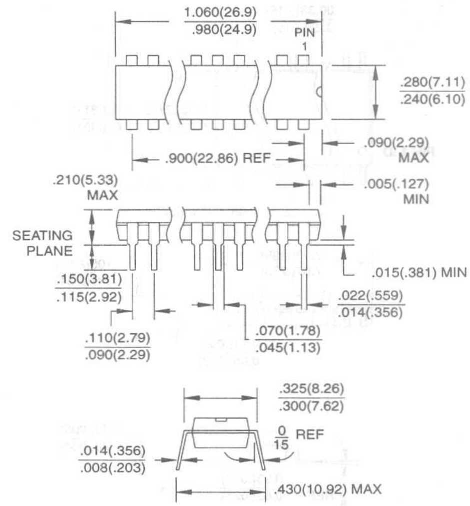


*Ceramic lid standard unless specified.

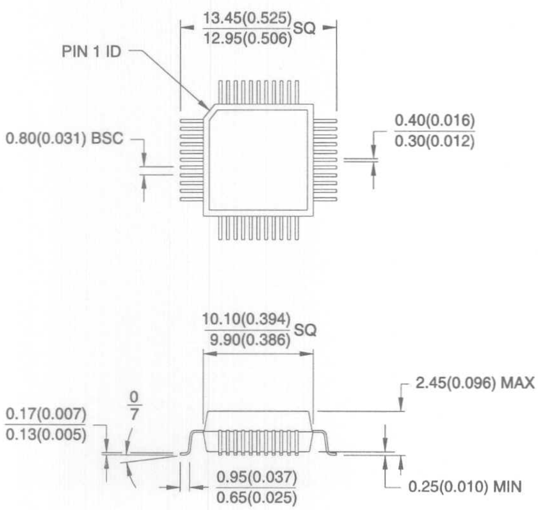
40P6, 40 Lead, 0.600" Wide,
Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AC



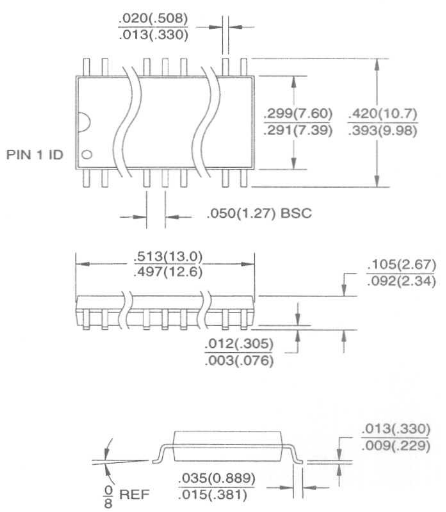
20P3, 20 Lead, 0.300" Wide,
Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-011 AD



44Q, 44 Lead,
Plastic Gull Wing QuadFlat Package (PQFP)
Dimensions in Inches and (Millimeters)

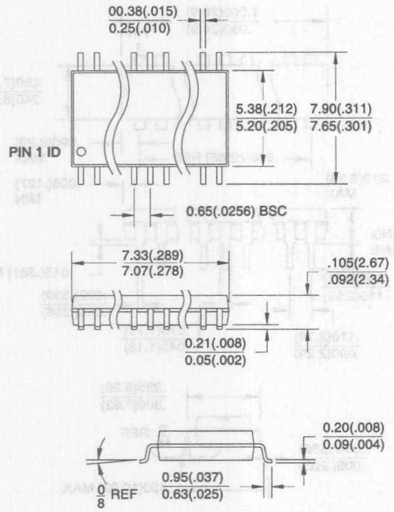


20S, 20 Lead, 0.300" Wide,
Plastic Gull Wing Small Outline (SOIC)
Dimensions in Inches and (Millimeters)



8

20Y, 20 Lead, 5.3 mm Wide,
Plastic Shrink Small Outline (SSOP)
Dimensions in Millimeters and (Inches)



ATMEL

Section 9 Miscellaneous Information

EPROMs

Part Number	Organization	Speeds	Description
Battery-Voltage™ (2.7V to 3.6V)			
AT27BV256	32K x 8	70-150 ns	256K bit, 2.7-Volt to 3.6-Volt
AT27BV512	64K x 8	90-150 ns	512K bit, 2.7-Volt to 3.6-Volt
AT27BV010	128K x 8	120-150 ns	1M bit, 2.7-Volt to 3.6-Volt EPROM
AT27BV1024	64K x 16	120-150 ns	1M bit, 2.7-Volt to 3.6-Volt
AT27BV020	256K x 8	120-150 ns	2M bit, 2.7-Volt to 3.6-Volt EPROM
AT27BV040	512K x 8	150 ns	4M bit, 2.7-Volt to 3.6-Volt EPROM
AT27BV4096	256K x 16	150 ns	4M bit, 2.7-Volt to 3.6-Volt
Low Voltage (3.0 to 3.6V)			
AT27LV256A	32K x 8	70-150 ns	256K bit, 3-Volt EPROM
AT27LV512A	64K x 8	90-150 ns	512K bit, 3-Volt EPROM
AT27LV010A	128K x 8	90-150 ns	1M bit, 3-Volt EPROM
AT27LV020A	256K x 8	120-150 ns	2M bit, 3-Volt EPROM
AT27LV040A	512K x 8	150 ns	4M bit, 3-Volt EPROM
Standard Voltage (5V)			
AT27C256R	32K x 8	45-150 ns	256K, 5-Volt EPROM
AT27C512R	64K x 8	45-150 ns	512K, 5-Volt EPROM
AT27C516	32K x 16	45-150 ns	512K, 5-Volt EPROM
AT27C010,L	128K x 8	45-150 ns	1M bit, 5-Volt EPROM Standard & Low Power
AT27C1024	64K x 16	45-150 ns	1M bit, 5-Volt EPROM
AT27C020	256K x 8	70-150 ns	2M bit, 5-Volt EPROM
AT27C2048	128K x 16	70-150 ns	2M bit, 5-Volt EPROM
AT27C040	512K x 8	80-150 ns	4M bit, 5-Volt EPROM
AT27C4096	256K x 16	85-150 ns	4M bit, 5-Volt EPROM
AT27C080	1024K x 8	100-150 ns	8M bit, 5-Volt EPROM

DataFlash™ (Serial-Interface Flash)

Part Number	Description
AT45D021	2M bit, 5-Volt Read and 5-Volt Write Serial-Interface Flash
AT45DB021	2M bit, 2.7-Volt Read and 2.7-Volt Write Serial-Interface Flash
AT45D041	4M bit, 5-Volt Read and 5-Volt Write Serial-Interface Flash
AT45DB041	4M bit, 2.7-Volt Read and 2.7-Volt Write Serial-Interface Flash
AT45D081	8M bit, 5-Volt Read and 5-Volt Write Serial-Interface Flash
AT45DB081	8M bit, 2.7-Volt Read and 2.7-Volt Write Serial-Interface Flash

Flash Memory Cards

Part Number	Organization	V_{CC}	Description
AT5FC001	1M byte	5-Volt	PCMCIA Compatible Flash Memory Card
AT5FC002	2M byte	5-Volt	PCMCIA Compatible Flash Memory Card
AT5FC004	4M byte	5-Volt	PCMCIA Compatible Flash Memory Card
AT5FC008	8M byte	5-Volt	PCMCIA Compatible Flash Memory Card

Flash Memories

Part Number	Organization	Speeds	Description
Battery-Voltage™ (2.7V to 3.6V)			
AT29BV010A	128K x 8	200-350 ns	1M bit, 2.7-Volt Read and 2.7-Volt Write, Sectored Flash
AT29BV020	256K x 8	250-350 ns	2M bit, 2.7-Volt Read and 2.7-Volt Write, Sectored Flash
AT29BV040A	512K x 8	250-350 ns	4M bit, 2.7-Volt Read and 2.7-Volt Write, Sectored Flash
AT49BV010	128K x 8	150-250 ns	1M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV020	256K x 8	200-250 ns	2M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV2048	128K x 16	150-250 ns	2M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV040	512K x 8	200-250 ns	4M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV4096	256K x 16	150-250 ns	4M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV080	1M x 8	150-250 ns	8M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
AT49BV8192	512K x 16	120-250 ns	8M bit, 2.7-Volt Read and 2.7-Volt Byte-Write Flash
Low Voltage (3V to 3.6V)			
AT29LV256	32K x 8	150-250 ns	256K, 3-Volt Read and 3-Volt Write, Sectored Flash
AT29LV512	64K x 8	150-250 ns	512K, 3-Volt Read and 3-Volt Write, Sectored Flash
AT29LV010A	128K x 8	150-250 ns	1M bit, 3-Volt Read and 3-Volt Write, Sectored Flash
AT29LV1024	64K x 16	150-250 ns	1M bit, 3-Volt Read and 3-Volt Write, Sectored Flash
AT29LV020	256K x 8	200-250 ns	2M bit, 3-Volt Read and 3-Volt Write, Sectored Flash
AT29LV040A	512K x 8	200-250 ns	4M bit, 3-Volt Read and 3-Volt Write, Sectored Flash
AT49LV010	128K x 8	150-250 ns	1M bit, 3-Volt Read and 3-Volt Byte-Write Flash
AT49LV020	256K x 8	150-250 ns	2M bit, 3-Volt Read and 3-Volt Byte-Write Flash
AT49LV2048	128K x 16	120-250 ns	2M bit, 3-Volt Read and 3-Volt Byte-Write Flash
AT49LV040	512K x 8	200-250 ns	4M bit, 3-Volt Read and 3-Volt Byte-Write Flash
AT49LV4096	256K x 16	120-250 ns	4M bit, 3-Volt Read and 3-Volt Byte-Write Flash
AT49LV8192	512K x 16	120-250 ns	8M bit, 3-Volt Read and 3-Volt Byte-Write Flash
Standard Voltage (5V)			
AT29C256	32K x 8	70-250 ns	256K, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C257	32K x 8	70-250 ns	256K, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C512	64K x 8	70-200 ns	512K, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C1024	64K x 16	70-200 ns	1M bit, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C010A	128K x 8	70-200 ns	1M bit, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C020	256K x 8	90-200 ns	2M bit, 5-Volt Read and 5-Volt Write, Sectored Flash
AT29C040A	512K x 8	120-250 ns	4M bit, 5-Volt Read and 5-Volt Write, Sectored Flash
AT49F010	128K x 8	70-120 ns	1M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F1025	64K x 16	70-120 ns	1M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F020	256K x 8	90-150 ns	2M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F2048	128K x 16	70-120 ns	2M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F040	512K x 8	90-150 ns	4M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F4096	256K x 16	90-120 ns	4M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F080	1M x 8	90-150 ns	8M bit, 5-Volt Read and 5-Volt Byte-Write Flash
AT49F8192	512K x 16	90-120 ns	8M bit, 5-Volt Read and 5-Volt Byte-Write Flash

FPGA Serial Configuration E^{2}PROM

Part Number	Memory Size	Description
AT17C65	65,536 x 1	65K FPGA Configuration E^{2}PROM
AT17C128	131,072 x 1	128K FPGA Configuration E^{2}PROM
AT17C256	262,144 x 1	256K FPGA Configuration E^{2}PROM
Low Voltage		
AT17LV65	65,536 x 1	65K FPGA Configuration E^{2}PROM, 3.3-Volt
AT17LV128	131,072 x 1	128K FPGA Configuration E^{2}PROM, 3.3-Volt
AT17LV256	262,144 x 1	256K FPGA Configuration E^{2}PROM, 3.3-Volt

Gate Arrays/Embedded Arrays

Device Name	Gates	Description
ATL35 Series	0K-2500K	0.35-Micron CMOS Gate Array/Embedded Array, 1.0-Volt to 3.3-Volt Operation, 16 Versions with Various Pin & Gate Count, Memory, Megacells
ATL50 Series	4K-1120K	0.5-Micron CMOS Gate Array/Embedded Array, 2.0-Volt & 3.3-Volt Mixed Voltage Operation, 16 Versions with Various Pin & Gate Counts, Memory, Megacells
ATLS60 Series	12.5K-150K	0.6-Micron CMOS Gate Array/Embedded Array, 3.3-Volt & 5.0-Volt Operation, Staggered Row Bond Pads, 8 Versions with Various Pin & Gate Counts, Memory, Megacells
ATL60 Series	4K-1120K	0.6-Micron CMOS Gate Array/Embedded Array, 3.3-Volt & 5.0-Volt Operation, 16 Versions with Various Pin & Gate Counts, Memory, Megacells
ATLV Series	2K-35K	1.0-Micron CMOS Gate Array/Embedded Array, 1.0-Volt & 3.3-Volt Operation, 8 Underlayers with Various Pin & Gate Counts, Memory, Megacells

Cell-Based ICs

Part Number	Description
ECPD07	0.8-Micron CMOS Cell-Based IC Family, 5.0-Volt Operation, Digital, Analog, Memory, Megacells
AT55K	0.5-Micron CMOS Cell-Based IC Family, 3.3-Volt Operation, Digital, Analog, Memory, Megacells
AT56K	0.35-Micron CMOS Cell-Based IC Family, 3.3-Volt Operation, Digital, Analog, Memory, Megacells
AT57K	0.25-Micron CMOS Cell-Based IC Family, 3.3-Volt Operation, Digital, Analog, Memory, Megacells
AT19.6K	0.8-Micron E^{2}PROM with Logic IC Family, 5.0-Volt Operation
AT19.7K	0.7-Micron CMOS with Embedded E^{2}PROM IC Family, 5.0-Volt Operation
AT19.9K	0.6-Micron E^{2}PROM with Logic IC Family, 5.0-Volt Operation
AT55.7K	0.5-Micron CMOS with Embedded E^{2}PROM IC Family, 3.3-Volt Operation
AT35K	0.4-Micron E^{2}PROM with Logic IC Family, 3.3-Volt Operation
AT33K	0.4-Micron Flash with Logic IC Family, 3.3-Volt Operation
Macrocells	ARM7, ARM7T, SPARC, AVR, OAK DSP, Ethernet MAC, AT8051, AT8237, AT8251, AT8254, AT8255, AT8259, AT82530, AT14818, AT16450, RAM, ROM, Flash, E^{2}PROM, PCI, SPI, PCMCIA, USB, Codec, A/D, D/A, OpAmp, Comp, OSC

Fast Logic™

Part Number	Packages	Speeds	Description
AT16244F	48-Pin TSSOP / SSOP	2.5 ns	16-Bit Buffer/Line Driver
AT16244G	48-Pin TSSOP / SSOP	2.0 ns	16-Bit Buffer/Line Driver
AT16245F	48-Pin TSSOP / SSOP	2.5 ns	16-Bit Bi-Directional Transceiver
AT16245G	48-Pin TSSOP / SSOP	2.0 ns	16-Bit Bi-Directional Transceiver
AT16373F	48-Pin TSSOP / SSOP	2.5 ns	16-Bit Transparent Latch
AT16373G	48-Pin TSSOP / SSOP	2.0 ns	16-Bit Transparent Latch
AT16646F	56-Pin TSSOP / SSOP	2.5 ns	16-Bit Tri-State Register
AT16646G	56-Pin TSSOP / SSOP	2.0 ns	16-Bit Tri-State Register

Programmable Logic Devices

Part Number	Packages	Speeds	Description
Flash-Based			
ATF16V8B	20-Pin	7.5-25 ns	8 FFs, 8 I/O Pins, Standard Power
ATF16V8BQ/BQL	20-Pin	10-25 ns	8 FFs, 8 I/O Pins, Quarter Power, Low Power
ATF16V8C	20-Pin	5-7.5 ns	8 FFs, 8 I/O Pins, Standard Power
ATF16V8CZ	20-Pin	12-15 ns	8 FFs, 8 I/O Pins, Zero Power
ATF20V8B	24, 28-Pin	7.5-25 ns	8 FFs, 8 I/O Pins, Standard Power
ATF20V8BQ/BQL	24, 28-Pin	10-25 ns	8 FFs, 8 I/O Pins, Quarter Power, Low Power
ATF20V8C	24, 28-Pin	5-7 ns	8 FFs, 8 I/O Pins, Standard Power
ATF20V8CZ	24, 28-Pin	12-15 ns	8 FFs, 8 I/O Pins, Zero Power
ATF22V10B	24, 28-Pin	7.5-25 ns	10 FFs, 10 I/O Pins, Standard Power
ATF22V10BQ/BQL	24, 28-Pin	15-25 ns	10 FFs, 10 I/O Pins, Quarter Power, Low Power
ATF22V10C	24, 28-Pin	5-10 ns	10 FFs, 10 I/O Pins, Standard Power
ATF22V10CZ	24, 28-Pin	12-15 ns	10 FFs, 10 I/O Pins, Zero Power
ATFV750C/CL	24, 28-Pin	7.5-25 ns	20 FFs, 10 I/O Pins, Standard & Low Power
ATF1500/L	44-Pin	7.5-25 ns	32 Macrocell, Standard & Low Power
ATF1500A/AL	44-Pin	7.5-25 ns	32 Macrocell, Standard & Low Power
ATF1508/L	68, 84, 100, 160-Pin	7.5-25 ns	128 Macrocell, Standard & Low Power
Low Voltage			
ATF16LV8C	20-Pin	10-15 ns	8 FFs, 8 I/O Pins, Low Voltage
ATF16LV8CZ	20-Pin	15-25 ns	8 FFs, 8 I/O Pins, Low Voltage, Zero Power
AT22LV10/L	24, 28-Pin	20-30 ns	10 FFs, 10 I/O Pins, Standard & Low Power
ATF1500ABV/ABVL	44-Pin	12-25 ns	32 FFs, 32 I/O Pins, Standard & Low Power
ATF22LV10	24, 28-Pin	10-15 ns	10 FFs, 10 I/O Pins, Low Voltage
ATF22LV10CZ	24, 28-Pin	15-25 ns	10 FFs, 10 I/O Pins, Low Voltage, Zero Power
5-Volt EPROM-Based			
AT22V10/L*	24, 28-Pin	15-25 ns	10 FFs, 10 I/O Pins, Standard & Low Power
ATV750/L	24, 28-Pin	20-25 ns	20 FFs, 10 I/O Pins, Standard & Low Power
ATV750B/BL	24, 28-Pin	7.5-25 ns	20 FFs, 10 I/O Pins, Standard & Low Power
ATV2500H/L	40, 44-Pin	25-35 ns	48 FFs, 24 I/O Pins, Standard & Low Power
ATV2500B/BL	44-Pin	12-20 ns	48 FFs, 24 I/O Pins, Standard & Low Power
ATV2500BQ/BQL	40, 44-Pin	20-25 ns	48 FFs, 24 I/O Pins, Quarter Power, Low Power

*In last-time buy mode, replace with ATF22V10B/BQ/BQL

Programmable Logic Device Design Software

Part Number	Description
ATDS1000PC	Atmel - CUPL
ATDS1100PC	Atmel - Synario Entry (Includes ABEL, Schematic Entry, Simulation)
ATDS1120PC	Atmel - Synario Verilog Simulation
ATDS1130PC	Atmel - Synario VHDL Synthesis

Parallel E²PROMs

Part Number	Organization	Speeds	Description
High Speed			
AT28HC64B	8K x 8	55-120 ns	64K E²PROM with 64-Byte Page & Software Data Protection
AT28HC256	32K x 8	70-120 ns	256K E²PROM with 64-Byte Page & Software Data Protection
AT28HC256E	32K x 8	70-120 ns	256K E²PROM with Extended Endurance Standard & Low Power
***Battery-Voltage*™ (2.7V to 3.6V)**			
AT28BV16	2K x 8	250-300 ns	16K E²PROM, 2.7-Volt
AT28BV64	8K x 8	300 ns	64K E²PROM, 2.7-Volt
AT28BV64B	8K x 8	200-250 ns	64K E²PROM with 64-Byte Page & Software Data Protection, 2.7-Volt
AT28BV256	32K x 8	200-250 ns	256K E²PROM with 64-Byte Page & Software Data Protection, 2.7-Volt
Low Voltage (3.0V to 3.6V)			
AT28LV128	16K x 8	200 ns	128K E²PROM with 64-Byte Page & Software Data Protection, 3.0-Volt
AT28LV010	128K x 8	200-250 ns	1M bit E²PROM with 128-Byte Page & Software Data Protection, 3.0-Volt
Standard Voltage (5V)			
AT28C16	2K x 8	150 ns	16K E²PROM
AT28C16E	2K x 8	150 ns	16K E²PROM with Extended Endurance & Fast Write
AT28C17	2K x 8	150 ns	16K E²PROM with Ready/Busy
AT28C17E	2K x 8	150 ns	16K E²PROM with Ready/Busy & Extended Endurance & Fast Write
AT28C64	8K x 8	120-250 ns	64K E²PROM
AT28C64E	8K x 8	120-250 ns	64K E²PROM with Extended Endurance & Fast Write
AT28C64X	8K x 8	120-250 ns	64K E²PROM without Ready/Busy
AT28C64B	8K x 8	150-250 ns	64K E²PROM with 64-Byte Page & Software Data Protection
AT28C256	32K x 8	150-250 ns	256K E²PROM with 64-Byte Page & Software Data Protection
AT28C256E	32K x 8	150-250 ns	256K E²PROM with Extended Endurance
AT28C010	128K x 8	120-250 ns	1M bit E²PROM with 128-Byte Page & Software Data Protection
AT28C010E	128K x 8	120-250 ns	1M bit E²PROM with 128-Byte Page & Extended Endurance & Software Data Protection
AT28C040	512K x 8	200-250 ns	4M bit E²PROM with 256-Byte Page & Software Data Protection

Secure Memory ICs

Part Number	Memory Size	Description
AT88SC101	1024 x 1	1K Serial E²PROM with Security, 1 Memory Zone, 1024 Bits
AT88SC102	1024 x 1	1K Serial E²PROM with Security, 2 Memory Zones, 512 Bits Each
AT88SC103	1536 x 1	1K Serial E²PROM with Security, 3 Memory Zones, 512 Bits Each
AT88SC1601	15,872 x 1	16K Serial E²PROM with Security, 1 Memory Zone, 15,872 Bits
AT88SC1604	15,968 x 1	16K Serial E²PROM with Security, 3 Memory Zones, 4096 Bits Each, and 1 Memory Zone, 3680 Bits
RF ID ASICs	Up to 16K x 1	Analog, Digital & Memory on Single-Chip ASIC

9

Serial E^{2}PROMs

Part Number	Organization	V_{CC}	Description
AT24C01	128 x 8	1.8, 2.5, 2.7, 5.0 V	1K, 2-Wire Bus Serial E^{2}PROM, Non-Cascadable
AT24C21	128 x 8	2.5 V	1K, 2-Wire Bus Serial E^{2}PROM, Dual Mode, Plug & Play Operation
AT24C01A	128 x 8	1.8, 2.5, 2.7, 5.0 V	1K, 2-Wire Bus Serial E^{2}PROM
AT24C02	256 x 8	1.8, 2.5, 2.7, 5.0 V	2K, 2-Wire Bus Serial E^{2}PROM
AT24C04	512 x 8	1.8, 2.5, 2.7, 5.0 V	4K, 2-Wire Bus Serial E^{2}PROM
AT24C08	1024 x 8	1.8, 2.5, 2.7, 5.0 V	8K, 2-Wire Bus Serial E^{2}PROM
AT24C16	2048 x 8	1.8, 2.5, 2.7, 5.0 V	16K, 2-Wire Bus Serial E^{2}PROM
AT24C164	2048 x 8	1.8, 2.5, 2.7, 5.0 V	16K, 2-Wire Bus Serial E^{2}PROM with Cascadable Feature
AT24C32	4096 x 8	1.8, 2.5, 2.7, 5.0 V	32K, 2-Wire Bus Serial E^{2}PROM with Cascadable Feature
AT24C64	8192 x 8	1.8, 2.5, 2.7, 5.0 V	64K, 2-Wire Bus Serial E^{2}PROM with Cascadable Feature
AT24C128	16,384 x 8	1.8, 2.7, 5.0 V	128K, 2-Wire Bus Serial E^{2}PROM with Cascadable Feature
AT24C256	32,768 x 8	1.8, 2.7, 5.0 V	256K, 2-Wire Bus Serial E^{2}PROM with Cascadable Feature
AT25010	128 x 8	1.8, 2.7, 5.0 V	1K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25020	256 x 8	1.8, 2.7, 5.0 V	2K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25040	512 x 8	1.8, 2.7, 5.0 V	4K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25080	1024 x 8	1.8, 2.7, 5.0 V	8K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25160	2048 x 8	1.8, 2.7, 5.0 V	16K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25320	4096 x 8	1.8, 2.7, 5.0 V	32K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25640	8192 x 8	1.8, 2.7, 5.0 V	64K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT25128	16,384 x 8	1.8, 2.7, 5.0 V	128K, SPI Bus Serial E^{2}PROM, Supports SPI Mode 0 and 3
AT93C46	64 x 16 / 128 x 8	1.8, 2.5, 2.7, 5.0 V	1K, 3-Wire Bus Serial E^{2}PROM
AT93C46A	64 x 16	1.8, 2.5, 2.7, 5.0 V	1K, 3-Wire Bus Serial E^{2}PROM
AT93C56	128 x 16 / 256 x 8	2.5, 2.7, 5.0 V	2K, 3-Wire Bus Serial E^{2}PROM
AT93C57	128 x 16 / 256 x 8	2.5, 2.7, 5.0 V	2K, 3-Wire Bus Serial E^{2}PROM with Special Address
AT93C66	256 x 16 / 512 x 8	2.5, 2.7, 5.0 V	4K, 3-Wire Bus Serial E^{2}PROM
AT59C11	64 x 16 / 128 x 8	2.5, 2.7, 5.0 V	1K, 4-Wire Bus Serial E^{2}PROM
AT59C22	128 x 16 / 256 x 8	2.5, 2.7, 5.0 V	2K, 4-Wire Bus Serial E^{2}PROM
AT59C13	256 x 16 / 512 x 8	2.5, 2.7, 5.0 V	4K, 4-Wire Bus Serial E^{2}PROM

E^2 Logic Family: E^{2}PROM + Gate Array

Part Number	E^{2}PROM Bits	Number of Usable Gates	Number of I/O	Package Type
AT88SC230	2,048	3,000	100	PQFP/TQFP/PLCC
AT88SC1610	16,384	1,500	8	PDIP/SOIC
AT88SC16350	16,384	35,000	144	PQFP/TQFP/PLCC

Microcontrollers

Part Number	Memory Size	Description
AT89C51	4K x 8	80C31 Microcontroller with 4K bytes Flash
AT89LV51	4K x 8	2.7-Volt, 80C31 Microcontroller with 4K bytes Flash
AT89C52	8K x 8	80C32 Microcontroller with 8K bytes Flash
AT89LV52	8K x 8	2.7-Volt, 80C32 Microcontroller with 8K bytes Flash
AT89C1051	1K x 8	80C31 Microcontroller with 1K bytes Flash, 20-Pin Package
AT89C2051	2K x 8	80C31 Microcontroller with 2K bytes Flash, 20-Pin Package
AT89S8252	8K x 8	Downloadable Microcontroller with 8K bytes Flash and 2K bytes E^2PROM
AT89S53	12K x 8	Downloadable Microcontroller with 12K bytes Flash
AT89C55	20K x 8	80C32 Microcontroller with 20K bytes Flash
AT90S1200	1K x 8	AVR® RISC Downloadable Microcontroller with 1K bytes Flash and 64 bytes E^2PROM, 20-Pin Package
AT90S2313	2K x 8	AVR® RISC Downloadable Microcontroller with 2K bytes Flash and 128 bytes E^2PROM, 20-Pin Package
AT90S8515	8K x 8	AVR® RISC Downloadable Microcontroller with 8K bytes Flash and 512 bytes E^2PROM, 40-Pin Package

AVR® Development Tools

Development Tool	Description
AT90ICE1200	Atmel In-Circuit Emulator
AT90DEV1200	Atmel Evaluation Development Board

Cache Logic® FPGAs

Part Number	Registers	Usable Gates	Frequency	Description
AT6002	1,024	6K	350 MHz	96 I/O Pins, 5-Volt, Very Low Power
AT6003	1,600	9K	350 MHz	120 I/O Pins, 5-Volt, Very Low Power
AT6005	3,136	15K	350 MHz	140 I/O Pins, 5-Volt, Very Low Power
AT6010	6,400	30K	350 MHz	204 I/O Pins, 5-Volt, Very Low Power
Low Voltage				
AT6002LV	1,024	6K	250 MHz	96 I/O Pins, 3-Volt, Very Low Power
AT6003LV	1,600	9K	250 MHz	120 I/O Pins, 3-Volt, Very Low Power
AT6005LV	3,136	15K	250 MHz	140 I/O Pins, 3-Volt, Very Low Power
AT6010LV	6,400	30K	250 MHz	204 I/O Pins, 3-Volt, Very Low Power

FPGA Design Development Software

FPGA design tools are available across a broad range of CAE tool vendors and PC and workstation platforms. Design methods supported include: schematic capture, logic synthesis (VHDL and Verilog), PLD entry, (ABEL and CUPL), and automatic component generation of hard macros for user-parametrized structured logic (arithmetic elements, counters, registers, encoders, decoders, and other common functions). Refer to current Configurable Logic Data Book.

CAE Tool Support:

Cadence, Everest, Exemplar, Intergraph, Mentor, OrCAD, Synopsys, Synario, Verilog, ViewLogic.

Platform Support:

PC, SUN Workstations, HP Workstations.